History-Preserving Bisimulations on Reversible Calculus of Communicating Systems

Clément Aubert, Ioana Cristescu

History-Preserving Bisimulations on Reversible Calculus of Communicating Systems

Clément Aubert
School of Computer and Cyber Sciences, Augusta University, USA
caubert@augusta.edu

Ioana Cristescu
Inria Rennes, France
ioana-domnina.cristescu@inria.fr

Abstract

History- and hereditary history-preserving bisimulation (HPB and HHPB) are classical equivalences relations for denotational models of concurrency that are meaningful to study reversible computation. Finding their counterpart in process algebras is an open problem, with some partial successes: there exists in Calculus of Communicating Systems (CCS) an equivalence based on causal trees that corresponds to HPB. In Reversible CSS (RCCS), there are bisimulations corresponding to HHPB, but they consider only restricted classes of process. We propose equivalences on RCCS processes that correspond to HPB *and* HHPB *on all processes*. The equivalences exploit not only reversibility but also the memory mechanism of RCCS, and our development shows why both are needed.

2012 ACM Subject Classification Theory of computation → Program semantics; Computing methodologies → Concurrent programming languages

Keywords and phrases Formal semantics, Process algebras and calculi, Configuration structures, Distributed and reversible computation, Auto-concurrency

Acknowledgements The authors would like to thank the reviewers of an earlier version of this work.

1 Introduction

Reversing Concurrent Computation Reversible systems have the possibility of backtracking to return to some previous state. Implementing reversibility in a programming language often requires a mechanism to record the history of the execution. Ideally, this history should be *complete*, so that every forward step can be backtracked, and *minimal*, so that only the relevant information is saved. Concurrent programming languages have additional requirements: the history should be *distributed*, to avoid centralization, and should prevent steps that required a sychronization with other parts of the program to backtrack without undoing this synchronization. To fulfill those requirements, Reversible Calculus of Communicating Systems (RCCS) [5, 6] uses *memories* attached to the threads of a process.

Equivalences for Reversible Processes A theory of reversible concurrent computation relies not only on a syntax, but also on "meaningful" behavioral equivalences. This paper studies behavioral equivalences defined on configuration structures [18], a classical denotational model for concurrency. In configuration structures, an *event* represents an execution step, and a *configuration*—a set of events that occurred—represents a state. A forward transition is then represented as moving from a configuration to one of its supersets. Backward transitions have a "built-in" representation: it suffices to move from a configuration to one of its subset. Multiple behavioral equivalences have been defined for configuration structures ; some of them, like *history-* and *hereditary history-preserving bisimulations* (HPB and HHPB) [2, 3, 8, 12, 15], use that "built-in" notion of reversibility.

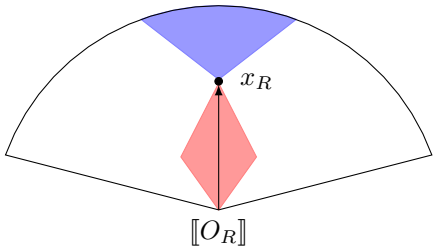


Figure 1 The encoding of O_R, of the future and of the past of a reversible process R.

Encoding Reversible Processes in Configuration Structures An ongoing research effort [1, 11] is to transfer equivalences defined on denotational models, which are by construction adapted for reversibility, back into reversible process algebras. Of course, showing that an equivalence on configuration structures corresponds to one on RCCS processes depends on the encoding of RCCS processes into configuration structures. A previously developed [1] encoding considered only *reachable* reversible processes: a process R is reachable if it can backtrack to a process O_R with an empty memory, called its *origin*, that corresponds to its state before the execution started. Then in the configuration structure $[\![O_R]\!]$, obtained using the common mapping for CCS processes [18], a configuration x_R is identified corresponding to the current state of R. In this set-up, the encoding of R is *one* configuration x_R, in $[\![O_R]\!]$: every configuration "below" is the "past" of R, every configuration "above", its "future" (Fig. 1).

Contribution This paper improves on previous results [1, 10, 12] by defining relations on RCCS processes that correspond to HPB *and* HHPB on *all* processes. We introduce an encoding of memories into *identified configuration structures*, an extension to configuratin structures. This encoding is independent from the rest of the process and we show that, as expected, the "past" of a process corresponds to the encoding of its memory. The memories attached to a process are no longer only a syntactic layer to implement reversibility, but become essential to define equivalences. This result gives an insight on the expressiveness of reversibility, as the back-and-forth moves of a process are not enough to capture HHPB.

Related work The correspondences between HHPB and back-and-forth bisimulations for restricted classes of processes [1, 10] inspired some of the work presented here. Our approach shares similarity with causal trees—in the sense that we encode only *part of the execution*, its "past", in a denotational representation—where some bisimulation corresponds to HPB [7]. To some extent, HPB and HHPB are considered "the gold standard" for measuring equivalence classes on configuration structures. However, no relation on labeled transition systems (being CCS, RCCS, or CSSK [9, 11]) was known to capture it. We summarize the current state of knowledge in Fig. 6.

Outline & Prerequisite We start by recalling the definitions of configuration structures (Sect. 2.1), of the encoding of CCS terms in configuration structures (Sect. 2.2), and of RCCS (Sect. 2.3). We then recall the definitions of (hereditary) history-preserving bisimulations on configuration structures (Sect. 3.1) and introduce our definition of back-and-forth bisimulation on RCCS that we use to recall previous results (Sect. 3.2). The new material is presented in Sect. 4, which starts by defining identified configuration structures (Sect. 4.1). Sect. 4.2 defines and illustrates with examples how identified configuration structures can encode

memories, and state some properties about them. Finally, Sect. 4.3 uses this encoding to define relations on RCCS processes that are then proven to correspond to HPB and HHPB on configuration structures and connects them to our back-and-forth bisimulation, and Sect. 5 concludes with an interesting connection to our previous formalism.

Sect. 2 is compact and serves more as a reminder than as an actual introduction to the topic at stake, but the relations under study as well as RCCS are introduced in great detail and with numerous examples that should guide the reader. As it is common in the study of concurrent computation [17, 19], we use category theory as a "common language". Only the notion of isomorphism, that is recalled in the body of our work, is really needed, but the curious reader can find a more rigorous development in Appendix A. All the proofs and some auxiliary results are gathered in Appendix B, but we tried to give insights and intuitions on how, and why, the results hold.

2 Preliminary Definitions

We recall the definitions of configuration structures and some of their restrictions (Sect. 2.1), and how to encode CCS processes into configuration structures (Sect. 2.2). Reversible CCS is recalled last (Sect. 2.3).

We write $\subseteq$ the set inclusion, $\mathcal{P}$ the power set, $\setminus$ the set difference, $A \to B$ (resp. $A \rightharpoonup B$) the set of (resp. partial) functions between A and B, $f\restriction_C$ the restriction of $f : A \to B$ to $C \subseteq A$, $f \cup \{a \mapsto b\}$ the function defined as $f : A \to B$ on A that additionally maps $a \notin A$ to b, and inversely we write $f \setminus \{a \mapsto b\}$ if $f : A \to B$ and $f(a) = b$ for $f\restriction_{A\setminus\{a\}}$.

Let $\mathsf{N} = \{a, b, c, \dots\}$ be a set of *names* and $\overline{\mathsf{N}} = \{\overline{a}, \overline{b}, \overline{c}, \dots\}$ its set of *co-names*. The *complement* of a (co-)name is given by a bijection $\bar{\cdot} : \mathsf{N} \to \overline{\mathsf{N}}$, whose inverse is also written $\bar{\cdot}$. We write $\overrightarrow{a}$ for a list of names $a_1, \cdots, a_n$. We define the set of labels $\mathsf{L} = \mathsf{N} \cup \overline{\mathsf{N}} \cup \{\tau\}$, and use α (resp. λ, ν, μ) to range over L (resp. $\mathsf{L}\setminus\{\tau\}$).

2.1 Configuration Structures

▶ **Definition 1** (Configuration structure). *A* configuration structure $\mathcal{C}$ *is a tuple* (E, C, L, ℓ) *where* E *is a set of events,* L *is a set of labels,* $\ell : E \to L$ *is a labeling function and* $C \subseteq \mathcal{P}(E)$ *is a set of subsets satisfying:*

$$\forall x \in C, \forall e \in x, \exists z \in C \text{ finite}, e \in z, z \subseteq x \qquad \text{(Finiteness)}$$

$$\forall x \in C, \forall d, e \in x, d \neq e \Rightarrow \exists z \in C, z \subseteq x, d \in z \iff e \notin z \qquad \text{(Coincidence Freeness)}$$

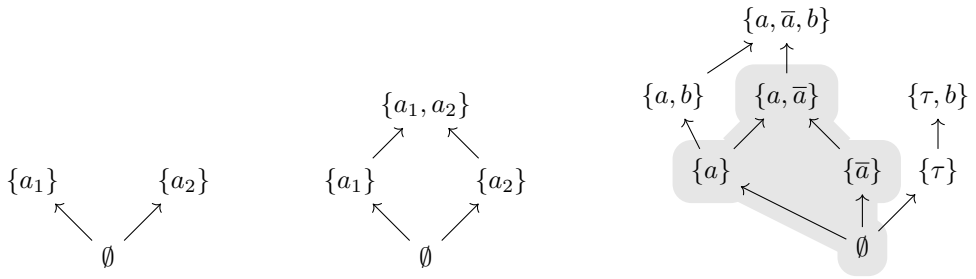
$$\forall X \subseteq C, \exists y \in C \text{ finite}, \forall x \in X, x \subseteq y \Rightarrow \bigcup X \in C \qquad \text{(Finite Completeness)}$$

$$\forall x, y \in C, x \cup y \in C \Rightarrow x \cap y \in C \qquad \text{(Stability)}$$

We denote $\mathbf{0}$ *the configuration structure with* $E = \emptyset$*, and write* $x \xrightarrow{e} y$ *and* $y \overset{e}{\rightsquigarrow} x$ *for* $x, y \in C$ *such that* $x = y \cup \{e\}$*.*

For the rest of this paper, let x, y, z range over configurations, d, e range over events, and assume that we are always given $\mathcal{C} = (E, C, L, \ell)$ and $\mathcal{C}_i = (E_i, C_i, L_i, \ell_i)$, for $i = 1, 2$.

▶ **Definition 2** (Causality, concurrency, and maximality). *For* $x \in C$ *and* $d, e \in x$*, the* causality relation on x *is given by* $d <_x e$ *iff* $d \leqslant_x e$ *and* $d \neq e$*, where* $d \leqslant_x e$ *iff for all* $y \in C$ *with* $y \subseteq x$*, we have* $e \in y \Rightarrow d \in y$*. If, for every* $x \in C$ *such that* $d, e \in x$*, we have* $d <_x e$*, then we write* $d < e$*. The* concurrency relation on x *is given by* $d \operatorname{co}_x e$ *iff* $\neg(d <_x e \vee e <_x d)$*. Finally,* x *is* maximal *if* $\forall y \in C$*,* $x = y$ *or* $x \not\subseteq y$*.*



(a) $[\![a + a]\!]$ **(b)** $[\![a \mid a]\!]$ **(c)** $[\![\overline{a} \mid a.b]\!]$

Figure 2 Examples of configuration structures encoding CCS processes

▶ **Definition 3** (Generation of a structure from a configuration)**.** *For $x \in C$, the* configuration structure generated by x *is* $x\downarrow = (x, \{y \mid y \in C, y \subseteq x\}, \{\alpha \mid \exists e \in x, \ell(e) = \alpha\}, \ell\restriction_x)$[1].

▶ **Example 4.** Consider the configuration structures of Fig. 2, where we make the abuse of notation of writing the events as their labels (with a subscript if multiple events have the same label), and where the sets of events, of configurations and of labels can be read from the diagram. Note that two events with complementary names can happen at the same time (Fig. 2c), in which case they are labeled with τ, as is usual in CCS. In Fig. 2c, we have $a < b$ and $\tau < b$; and in Fig. 2b, a_1 and a_2 are concurrent in the configuration $\{a_1, a_2\}$. A configuration structure can have one (Fig. 2b) or multiple (Fig. 2a and 2c) maximal configurations. Finally, we grayed out $\{a, \overline{a}\}\downarrow$ in Fig. 2c.

We now recall how process algebra constructors are defined on configuration structures [17]. The definition below may seem technical, but the encoding of CCS processes into configuration structures (Definition 8) will make it clear that it captures the right notions. Product, relabeling and restriction are needed only to define parallel composition.

This definition uses the product $(\times_\star, p_1, p_2)$ of the category which has sets as objects and partial functions as morphisms [17, Appendix A]: letting $\star$ denote *undefined* for a partial function and $C^\star = C \cup \{\star\}$ for a set C, we define, for two sets A and B,

$$A \times_\star B = \{(a, \star) \mid a \in A\} \cup \{(\star, b) \mid b \in B\} \cup \{(a, b) \mid a \in A, b \in B\}$$

with $p_1 : A \times_\star B \to A^\star$ and $p_2 : A \times_\star B \to B^\star$ the two projections.

▶ **Definition 5** (Operations on configuration structures [1, 14])**.**
The product *of $\mathcal{C}_1$ and $\mathcal{C}_2$ is $\mathcal{C}_1 \times \mathcal{C}_2 = (E_1 \times_\star E_2, C, L, \ell)$. Define the projections $\pi_i : \mathcal{C} \to \mathcal{C}_i$ and the configurations $x \in C$ such that:*

$$\forall e \in E, \pi_i(e) = p_i(e), \pi_i(\ell_i(e)) = \ell_i(\pi_i(e))$$
$$\pi_i(x) \in C_i, \text{ with } \pi_i(x) = \{e_i \mid \pi_i(e) = e_i \neq \star \text{ and } e \in x\}$$
$$\forall e, e' \in x, \pi_1(e) = \pi_1(e') \neq \star \text{ or } \pi_2(e) = \pi_2(e') \neq \star \Rightarrow e = e'$$
$$\forall e \in x, \exists z \subseteq x \text{ finite}, \pi_1(x) \in C_1, \pi_2(x) \in C_2, e \in z$$
$$\forall e, e' \in x, e \neq e' \Rightarrow \exists z \subseteq x, \pi_i(z) \in C_i, e \in z \iff e' \notin z$$

[1] For the reader familiar with event structures, a configuration x defines an event structure $(x, \leqslant_x, \ell)$. The construction here mirrors the transformation from an event structure to a configuration structure [19].

The labeling function $\ell : E_1 \times_\star E_2 \to L = L_1 \cup L_2 \cup (L_1 \times L_2)$ *is*

$$\ell(e) = \begin{cases} \ell_1(e_1) & \text{if } \pi_1(e) = e_1 \neq \star \text{ and } \pi_2(e) = \star \\ \ell_2(e_2) & \text{if } \pi_1(e) = \star \text{ and } \pi_2(e) = e_2 \neq \star \\ (\ell_1(e_1), \ell_2(e_2)) & \text{otherwise} \end{cases}$$

The relabeling *of* $\mathcal{C}_1$ *along* $r : E_1 \to L$ *is* $r \circ \mathcal{C}_1 = (E_1, C_1, L, r)$.

The restriction of a set of events $A \subseteq E_1$ *in* $\mathcal{C}_1$ *is* $\mathcal{C}_1 \restriction_A = (E_1 \setminus A, C, L, \ell_1 \restriction_{E_1 \setminus A})$, *where* $x \in C \iff x \in C_1$ *and* $x \cap A = \emptyset$, *and* $L = \{\alpha \mid \exists e \in A, \ell_1(e) = \alpha\}$.

The restriction of a name a *in* $\mathcal{C}_1$ *is* $\mathcal{C}_1 \restriction_a = \mathcal{C}_1 \restriction_{E_1^a}$ *where* $E_1^a = \{e \in E_1 \mid \ell(e) \in \{a, \overline{a}\}\}$. *For* $\overrightarrow{a} = a_1, \ldots, a_n$ *a list of names, we define similarly* $\mathcal{C}_1 \restriction_{\overrightarrow{a}} = \mathcal{C}_1 \restriction_{E_1^{\overrightarrow{a}}}$ *for* $E_1^{\overrightarrow{a}} = \{e \in E_1 \mid \ell(e) \in \{a_1, \overline{a_1}, \ldots, a_n, \overline{a_n}\}\}$.

The parallel composition *of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *is* $\mathcal{C}_1 \mid \mathcal{C}_2 = \big(r \circ (\mathcal{C}_1 \times \mathcal{C}_2)\big) \restriction_{E_3^\perp}$, *with*

- $\mathcal{C}_1 \times \mathcal{C}_2 = \mathcal{C}_3 = (E_3, C_3, L_3, \ell_3)$ *is the product;*
- $r \circ \mathcal{C}_3$ *with* $r : E_3 \to L_1 \cup L_2 \cup \{\perp\}$ *defined as follows:*

$$r(e) = \begin{cases} \ell_3(e) & \text{if } \ell_3(e) \in \mathsf{N} \cup \overline{\mathsf{N}} \\ \tau & \text{if } \ell_3(e) \in \{(a, \overline{a}), (\overline{a}, a), \tau \mid a \in \mathsf{N}\} \\ \perp & \text{otherwise} \end{cases}$$

- $(r \circ \mathcal{C}_3) \restriction_{E_3^\perp}$, *where* $E_3^\perp = \{e \in E_3 \mid r(e) = \perp\}$.

The coproduct *of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *is* $\mathcal{C}_1 + \mathcal{C}_2 = \mathcal{C}$, *where* $E = (\{1\} \times E_1) \cup (\{2\} \times E_2)$ *and* $C = \{\{1\} \times x \mid x \in C_1\} \cup \{\{2\} \times x \mid x \in C_2\}$. *The labeling function* ℓ *is defined, for* $\pi_1(e) = i$, *as* $\ell(e) = \ell_i(\pi_2(e))$ *and* $L = L_1 \cup L_2$.

The prefixing *of* $\mathcal{C}_1$ *by the name* λ *is* $\lambda.\mathcal{C}_1 = (e \cup E_1, C, L_1 \cup \{\lambda\}, \ell)$, *for* $e \notin E_1$, *where* $x \in C \iff x = \emptyset \vee \exists x' \in C_1, x = x' \cup e$ *;* $\ell(e) = \lambda$ *and* $\forall e' \neq e, \ell(e') = \ell_1(e')$.

As detailed in Appendix A, configuration structures and "structure-preserving" functions form a category. This development can be omitted, except for the notion of isomorphisms:

▶ **Definition 6.** *We write* $\mathcal{C}_1 \cong \mathcal{C}_2$ *if there exists an isomorphism* $f = (f_E, f_L, f_C)$*; a bijection such that* $f_E : E_1 \to E_2$ *preserves labels,* $\ell_2(f_E(e)) = f_L(\ell_1(e))$, *for* $f_L : L_1 \to L_2$*; and* $f_C : C_1 \to C_2$ *is defined as* $f_C(x) = \{f_E(e) : e \in x\}$.

Some constraints were used in showing equivalences between process algebra and configuration structures:

▶ **Definition 7** (Constraints on configuration structures)**.** *If* $\forall x \in C$,

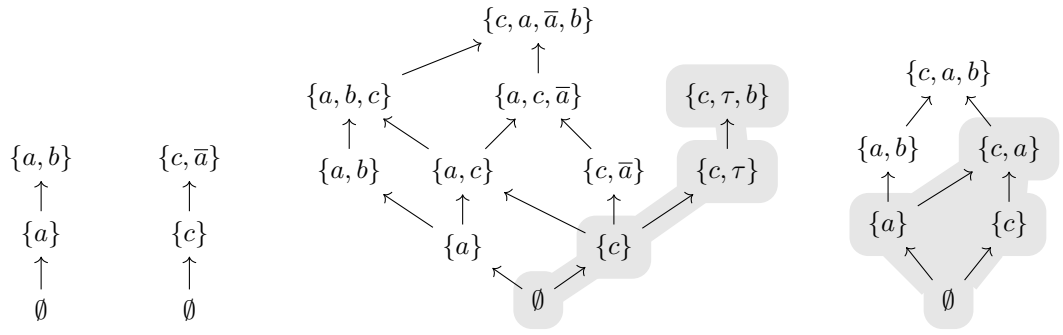
- $\forall e_1, e_2 \in x$, $\ell(e_1) = \ell(e_2)$ *implies* $e_1 = e_2$, *then* $\mathcal{C}$ *is* non-repeating *[10, Definition 3.5].*
- $\forall e_1, e_2 \in x$, $e_1 \operatorname{co}_x e_2$ *and* $\ell(e_1) = \ell(e_2)$ *implies* $e_1 = e_2$, *then* $\mathcal{C}$ *is* without auto-concurrency *[16, Definition 9.5].*
- $\forall e, e' \notin x$, $\big(x \xrightarrow{e} y, x \xrightarrow{e'} y', \ell(e) = \ell(e')$ *and* $\mathcal{C} \setminus y \cong \mathcal{C} \setminus y'\big)$ *implies* $e = e'$, *where* $\mathcal{C} \setminus z = (E \setminus z, \{z' \mid z' \cup z \in C\}, \ell \restriction_{E \setminus z})$, *then* $\mathcal{C}$ *is* singly labeled *[1, Definition 26].*

2.2 CCS and its Encoding in Configuration Structures

The set of CCS processes is inductively defined:

$$P, Q \coloneqq P \mid Q \parallel \lambda.P + \nu.Q \parallel P \backslash a \parallel 0 \qquad \text{(CCS Processes)}$$

We often omit 0, and write e.g. $a \mid (b + \overline{c})$ for $a.0 \mid (b.0 + \overline{c}.0)$. We work up to the structural congruence $\equiv$ of CCS–that we suppose familiar to the reader–, and write e.g. $P_1 \mid P_2 \mid P_3$ without parenthesis since $(P_1 \mid P_2) \mid P_3 \equiv P_1 \mid (P_2 \mid P_3)$. Finally, alpha-equivalence is written $=_\alpha$ and supposed familiar as well.



(a) $[\![a.b]\!]$ **(b)** $[\![c.\overline{a}]\!]$ **(c)** $[\![(a.b) \mid (c.\overline{a})]\!]$ **(d)** $[\![a.b \mid c]\!]$

Figure 3 (Identified) Configurations structures of Examples 9 and 28.

▶ **Definition 8** (Encoding a CCS process [19]). *Given a CCS process P, its encoding $[\![P]\!]$ as a configuration structure is built inductively:*

$$[\![\lambda.P + \nu.Q]\!] = [\![\lambda.P]\!] + [\![\nu.Q]\!] \qquad [\![\lambda.P]\!] = \lambda.[\![P]\!] \qquad [\![P\backslash a]\!] = [\![P]\!]\restriction_a$$
$$[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!] \qquad [\![0]\!] = \mathbf{0}$$

▶ **Example 9.** In addition to the examples of Fig. 2, the encoding of the processes $a.b$, $c.\overline{a}$, and their parallel composition $(a.b) \mid (c.\overline{a})$, as well as $a.b \mid c$, are given in Fig. 3, where the grayed out parts are $\{c, \tau, b\}\downarrow$ and $\{c, a\}\downarrow$.

▶ **Definition 10** (Restrictions on CCS processes). *A process P is* non-repeating *(resp.* without auto-concurrency*,* singly labeled*) if $[\![P]\!]$ is (Definition 7).*

▶ **Example 11.** Every non-repeating configuration is without auto-concurrency, but the other pairs of restrictions are independent, as suggested below:

	Non-repeating	**Without auto-concurrency**	**Singly labeled**
a	✓	✓	✓
$a.a$	✗	✓	✓
$(a.b) \mid a$	✗	✗	✓
$a + a$	✓	✓	✗
$a.(a + a)$	✗	✓	✗
$a \mid (a.(b + b))$	✗	✗	✗

2.3 Reversible CCS

Let I be a set of identifiers, and i, j range over its elements: an identifier can be anything, but a reasonable implementation would use integers or Linux's `pid_t` datatype. The set of *reversible* processes R is built on top of the set of CCS processes by adding memory stacks to the threads:

$$e ::= \langle i, \lambda, P\rangle \qquad \text{(Memory Events)}$$
$$m ::= \emptyset \parallel \curlyvee.m \parallel e.m \qquad \text{(Memory Stacks)}$$
$$T ::= m \rhd P \qquad \text{(Reversible Thread)}$$
$$R, S ::= T \parallel R \mid S \parallel R\backslash a \qquad \text{(RCCS Processes)}$$

We will treat CCS and "memory-less" reversible processes to be the same, i.e. we will sometimes treat P as being identical to $\emptyset \rhd P$: we come back to this in Definition 15.

$$i \notin \mathsf{I}(m) \ \frac{}{(m \rhd \lambda.P + Q) \xrightarrow{i:\lambda} \langle i, \lambda, Q\rangle.m \rhd P} \text{ act.} \qquad \frac{R \xrightarrow{i:\lambda} R' \quad S \xrightarrow{i:\overline{\lambda}} S'}{R \mid S \xrightarrow{i:\tau} R' \mid S'} \text{ syn.}$$

$$i \notin \mathsf{I}(m) \ \frac{}{\langle i, \lambda, Q\rangle.m \rhd P \overset{i:\lambda}{\rightsquigarrow} m \rhd (\lambda.P + Q)} \text{ act.}_* \qquad \frac{R \overset{i:\lambda}{\rightsquigarrow} R' \quad S \overset{i:\overline{\lambda}}{\rightsquigarrow} S'}{R \mid S \overset{i:\tau}{\rightsquigarrow} R' \mid S'} \text{ syn.}_*$$

$$i \notin \mathsf{I}(S) \ \frac{R \xrightarrow{i:\alpha}\!\!\!\!\rightarrow R'}{R \mid S \xrightarrow{i:\alpha}\!\!\!\!\rightarrow R' \mid S} \text{ par.} \qquad \frac{R \xrightarrow{i:\alpha}\!\!\!\!\rightarrow R' \quad a \notin \alpha}{R\backslash a \xrightarrow{i:\alpha}\!\!\!\!\rightarrow R'\backslash a} \text{ res.} \qquad \frac{R_1 \equiv R \xrightarrow{i:\alpha}\!\!\!\!\rightarrow R' \equiv R_1'}{R_1 \xrightarrow{i:\alpha}\!\!\!\!\rightarrow R_1'} \equiv$$

Figure 4 Rules of the labeled transition system

▶ **Definition 12** (Sets of identifiers and names). *We denote* $\mathsf{I}(e)$ *(resp* $\mathsf{I}(m)$*,* $\mathsf{I}(R)$*) the set of identifiers occurring in* e *(resp.* m*,* R*), and always take* $\mathsf{I} = \mathbb{N}$*. For a memory* m*, we let* $\mathrm{nm}(m) = \{\alpha \mid \alpha \in \mathsf{N} \text{ or } \overline{\alpha} \in \overline{\mathsf{N}} \text{ occurs in } m\}$ *be the set of (co-)names occurring in* m*.*

▶ **Definition 13** (Structural equivalence [1, Definition 5]). *Structural equivalence on* R *is the smallest equivalence relation generated by the following rules:*

$$m \rhd (\lambda.P + \nu.Q) \equiv m \rhd (\nu.Q + \lambda.P) \qquad \text{(Sum Symmetry)}$$
$$m \rhd ((\lambda.P + \nu.Q) + \mu.R) \equiv m \rhd (\lambda.P + (\nu.Q + \mu.R)) \qquad \text{(Sum Associativity)}$$
$$\frac{P =_\alpha Q}{m \rhd P \equiv m \rhd Q} \qquad (\alpha\text{-Conversion})$$
$$m \rhd (P \mid Q) \equiv (\Upsilon.m \rhd P \mid \Upsilon.m \rhd Q) \qquad \text{(Distribution of Memory)}$$
$$m \rhd P\backslash a \equiv (m \rhd P)\backslash a \text{ with } a \notin \mathrm{nm}(m) \qquad \text{(Scope of Restriction)}$$

The labeled transition system for RCCS is given by the rules of Fig. 4. We use $\xrightarrow{i:\alpha}\!\!\!\!\rightarrow$ for the union of $\xrightarrow{i:\alpha}$ (forward) and $\overset{i:\alpha}{\rightsquigarrow}$ (backward transition), and if there are indices $i_1, \ldots, i_n$ and labels $\alpha_1, \ldots, \alpha_n$ such that $R_1 \xrightarrow{i_1:\alpha_1}\!\!\!\!\rightarrow \cdots \xrightarrow{i_n:\alpha_n}\!\!\!\!\rightarrow R_n$, then we write $R_1 \xrightarrow{\star}\!\!\!\!\rightarrow R_n$.

Note that those rules are presented with the guarded sum of CCS for concision and clarity, but we will consider for instance $\emptyset \rhd a.P \xrightarrow{1:a} \langle 1, a, 0\rangle.\emptyset \rhd P$ to be a legal transition.

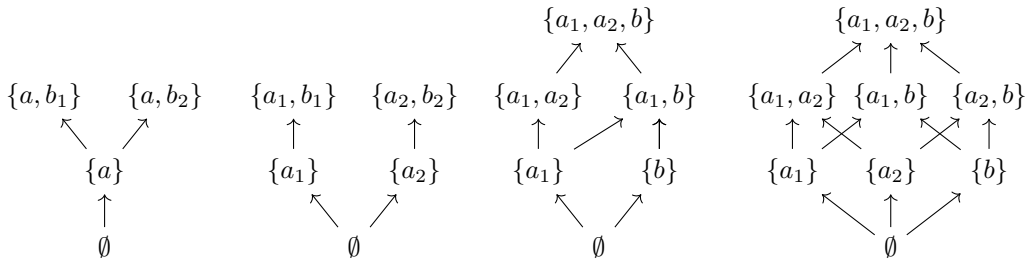
▶ **Example 14.** An example of valid (forward-only) execution, or trace, is:

$$\begin{aligned}
\emptyset \rhd (a.b \mid c.\overline{a}) &\equiv (\Upsilon.\emptyset \rhd a.b) \mid (\Upsilon.\emptyset \rhd c.\overline{a}) && \text{(Distribution of Memory)}\\
&\xrightarrow{1:c} (\Upsilon.\emptyset \rhd a.b) \mid (\langle 1, c, 0\rangle.\ \Upsilon\ .\emptyset \rhd \overline{a}) && \text{(act.)}\\
&\xrightarrow{2:\tau} (\langle 2, a, 0\rangle.\ \Upsilon\ .\emptyset \rhd b) \mid (\langle 2, \overline{a}, 0\rangle.\langle 1, c, 0\rangle.\ \Upsilon\ .\emptyset \rhd 0) && \text{(syn.)}\\
&\xrightarrow{3:b} (\langle 3, b, 0\rangle.\langle 2, a, 0\rangle.\ \Upsilon\ .\emptyset \rhd 0) \mid (\langle 2, \overline{a}, 0\rangle.\langle 1, c, 0\rangle.\ \Upsilon\ .\emptyset \rhd 0) && \text{(act.)}
\end{aligned}$$

Reading it from end to beginning and replacing $\xrightarrow{\vdots}$ with $\overset{\vdots}{\rightsquigarrow}$ gives a *backward-only* execution. Of course, an execution can be a mix of forward and backward transitions.

▶ **Definition 15** (Reachable [1, Lemma 1]). *If there is a CCS term* P *such that* $\emptyset \rhd P \xrightarrow{\star}\!\!\!\!\rightarrow R$*, we say that* R *is* reachable*, that* P *is the* origin *of* R *and write* $P = O_R$*.*

An important result [5, Lemma 10] furthermore states that this transition $\emptyset \rhd P \xrightarrow{\star}\!\!\!\!\rightarrow R$ can always be taken to be forward-only. Also, note that multiple RCCS terms can have the same origin, but that a reachable RCCS term has one unique origin (up to structural



(a) $[\![a.(b+b)]\!]$ **(b)** $[\![(a.b)+(a.b)]\!]$ **(c)** $[\![(a.a) \mid b]\!]$ **(d)** $[\![a \mid a \mid b]\!]$

Figure 5 Examples of configuration structures that are or not in HPB and HHPB relations

equivalence). We consider only reachable terms: unreachable terms are "dysfunctional", since their memory is not coherent [6] and they can not be "rewinded" back to an origin process.

▶ **Definition 16** (Restrictions on RCCS processes). *A process R is* non-repeating *(resp.* without auto-concurrency*,* singly labeled*) if $[\![O_R]\!]$ is (Definition 7).*

3 Reversible Bisimulations

In this section we introduce bisimulations on configuration structures (Sect. 3.1) and on RCCS (Sect. 3.2), which aim at capturing reversibility.

3.1 History-Preserving Bisimulations in Configuration Structures

History-preserving bisimulation (HPB) [2, 12, 13] and hereditary history-preserving bisimulation (HHPB) [2, 3] are equivalences on configuration structures that use label- and order-preserving bijections between the events of the two configuration structures.

▶ **Definition 17** (Label- and order-preserving functions). *A function $f : x_1 \to x_2$, for $x_i \in C_i$, $i \in \{1,2\}$ is* label-preserving *if $\ell_1(e) = \ell_2(f(e))$ for all $e \in x_1$. It is* order-preserving *if $e_1 \leqslant_{x_1} e_2 \Rightarrow f(e_1) \leqslant_{x_2} f(e_2)$, for all $e_1, e_2 \in x_1$.*

▶ **Definition 18** (HPB and HHPB). *A relation $\mathcal{R} \subseteq C_1 \times C_2 \times (E_1 \rightharpoonup E_2)$ such that $(\emptyset,\emptyset,\emptyset) \in \mathcal{R}$, and if $(x_1, x_2, f) \in \mathcal{R}$, then f is a label- and order-preserving bijection between x_1 and x_2 and (1) and (2) (resp. (1–4)) hold is called a* history- *(resp.* hereditary history-*)* preserving bisimulation (HPB, resp. HHPB) between $\mathcal{C}_1$ and $\mathcal{C}_2$.

$$\forall y_1, x_1 \xrightarrow{e_1} y_1 \Rightarrow \exists y_2, g, x_2 \xrightarrow{e_2} y_2, g\restriction_{x_1} = f, (y_1, y_2, g) \in \mathcal{R} \tag{1}$$
$$\forall y_2, x_2 \xrightarrow{e_2} y_2 \Rightarrow \exists y_1, g, x_1 \xrightarrow{e_1} y_1, g\restriction_{x_1} = f, (y_1, y_2, g) \in \mathcal{R} \tag{2}$$
$$\forall y_1, x_1 \overset{e_1}{\rightsquigarrow} y_1 \Rightarrow \exists y_2, g, x_2 \overset{e_2}{\rightsquigarrow} y_2, g = f\restriction_{y_1}, (y_1, y_2, g) \in \mathcal{R} \tag{3}$$
$$\forall y_2, x_2 \overset{e_2}{\rightsquigarrow} y_2 \Rightarrow \exists y_1, g, x_1 \overset{e_1}{\rightsquigarrow} y_1, g = f\restriction_{y_1}, (y_1, y_2, g) \in \mathcal{R} \tag{4}$$

We write that C_1 and C_2 are (H)HPB *if there exists a* (H)HPB *relation between them.*

▶ **Example 19.** Note that HPB and HHPB are two different relations, and that HHPB is not the structural congruence of CCS: the classical example [16] of processes whose encodings are HPB but not HHPB is $(a \mid (b+c)) + (a \mid b) + ((a+c) \mid b)$ and $(a \mid (b+c)) + ((a+c) \mid b)$. The two processes $a.(b+b)$ and $(a.b)+(a.b)$ are not structurally congruent in CCS, but their encodings, presented in Fig. 5a and 5b, are HHPB.

1. **F-R bisimulation** (Definition 20) on **non-repeating** CCS processes (Definition 10) corresponds to HHPB on their encoding [10].
2. **Back-and-forth barbed congruence** on **singly labeled** CCS processes (Definition 10) corresponds to HHPB on their encoding [1].
3. **Back-and-forth bisimulation** (Definition 20) and **Hereditary history-preserving bisimulation** (resp. history-preserving bisimulation) (Definition 32) on **all** CCS processes correspond to HHPB (resp. HPB) on their encoding (Theorem 36, Corollary 38).

Figure 6 Syntactical characterizations of HHPB

3.2 Back-and-forth Bisimulations in Reversible CCS

Let us now review the preliminary attempts to capture HHPB as a relation on process algebra terms [1, 10], and present the relation we will be using. We use the setting of RCCS to present these results, and synthesize them, along with our new result, in Fig. 6.

▶ **Definition 20** (B&F, SB&F, FR bisimulations)**.** *A relation* $\mathcal{R} \subseteq \mathsf{R} \times \mathsf{R} \times (\mathsf{I} \rightharpoonup \mathsf{I})$ *such that* $(\emptyset \rhd P_1, \emptyset \rhd P_2, \emptyset) \in \mathcal{R}$ *and if* $(R_1, R_2, f) \in \mathcal{R}$*, then* f *is a bijection between* $\mathsf{I}(R_1)$ *and* $\mathsf{I}(R_1)$ *and (5–8) hold is called* a back-and-forth-bisimulation (B&F bisim) between R_1 and R_2*.*

$$\forall S_1, R_1 \xrightarrow{i:\alpha} S_1 \Rightarrow \exists S_2, g, R_2 \xrightarrow{j:\alpha} S_2, g = f \cup \{i \mapsto j\}, (S_1, S_2, g) \in \mathcal{R} \tag{5}$$

$$\forall S_2, R_2 \xrightarrow{i:\alpha} S_2 \Rightarrow \exists S_1, g, R_1 \xrightarrow{j:\alpha} S_1, g = f \cup \{i \mapsto j\}, (S_1, S_2, g) \in \mathcal{R} \tag{6}$$

$$\forall S_1, R_1 \overset{i:\alpha}{\rightsquigarrow} S_1 \Rightarrow \exists S_2, f, R_2 \overset{j:\alpha}{\rightsquigarrow} S_2, g = f \setminus \{i \mapsto j\}, (S_1, S_2, g) \in \mathcal{R} \tag{7}$$

$$\forall S_2, R_2 \overset{i:\alpha}{\rightsquigarrow} S_2 \Rightarrow \exists S_1, g, R_1 \overset{j:\alpha}{\rightsquigarrow} S_1, g = f \setminus \{i \mapsto j\}, (S_1, S_2, g) \in \mathcal{R} \tag{8}$$

If $f = \mathrm{id}$*, we call* $\mathcal{R}$ *a* forward-reverse bisimulation (FR bisim)*, and if we remove the requirements on* f*, we call* $\mathcal{R}$ *a* simple back-and-forth bisimulation (SB&F bisim)*. We write that* R_1 *and* R_2 *(resp.* P_1 *and* P_2*)* are B&F bisim *if there exists a* B&F bisim *relation between them (resp. between* $\emptyset \rhd P_1$ *and* $\emptyset \rhd P_2$*), and similarly for* FR bisim *and* SB&F bisim*.*

FR bisim [9, 10] was used to get the first result on capturing HHPB:

▶ **Theorem 21** ([10, Theorem 5.4])**.** *Two CCS processes with non-repeating events are in* FR bisim *iff their encoding are* HHPB*.*

However, back-and-forth bisimulations do not use in a meaningful way the identifiers when restricted to non-repeating processes, which are trivially without auto-concurrency:

▶ **Theorem 22.** *On the class of processes without auto-concurrency,* B&F bisim *and* SB&F bisim *are the same relations.*

The intuition behind this theorem is that since two concurrent transitions sharing the same label can not be fired at the same time, the process being without auto-concurrency, the identifiers do not add any information. The proof is easy for the forward transitions, and uses an order on the transitions enforced by causality for the backward traces.

▶ **Example 23.** Observe that the bisimulation relation obtained from Definition 20 by only considering (5) and (6) and ignoring the identifiers is the "standard" bisimulation of CCS. The processes $(a.a) \mid b$ and $a \mid a \mid b$ are in this "standard" bisimulation, but their encodings, presented in Fig. 5c and 5d, are not HPB. Also note that those two processes are SB&F bisim but that their encodings are not HHPB. This example shows that SB&F bisim does

not characterize HHPB on processes with auto-concurrency, and supports that both the bijection on identifiers and the backward transitions are necessary to capture HHPB.

A second attempt [1] to capture HHPB used a *back-and-forth barbed congruence* on RCCS processes which was proven to correspond to HHPB on their encoding for the class of singly-labeled processes (Definition 10). Even if this restriction may seem lighter than forbidding not non-repeating structures and easier to justify (it is a restriction relative to "the future" of the configurations), it is actually an orthogonal restriction to being non-repeating, and forbid processes as simple as $a+a$ (Example 11). We lift both restrictions in the following, by proving that B&F bisim captures HHPB on *all* processes.

4 Lifting the Restrictions

To prove that B&F bisim corresponds to HHPB on all processes (Corollary 38), we first define bisimulations on RCCS (Sect. 4.3) that are then proven to correspond to (H)HPB (Theorem 36). The relation that characterizes HHPB coincides with B&F bisim (Theorem 37). Those relations on RCCS use *identified configuration structures* (Sect. 4.1) to encode the memories of reversible processes (Sect. 4.2).

4.1 Identified Configuration Structures

Identified configuration structures arise from the observation that events can carry, on top of a label, an identifier reflecting the history of a reversible process, as recorded by its memory.

▶ **Definition 24** (Identified configuration structure)**.** *An* identified configuration structure*, or* $\mathcal{I}$-structure*,* $\mathcal{I} = (E, C, L, \ell, I, m)$ *is a configuration structure* $\mathcal{C} = (E, C, L, \ell)$ *endowed with a set of identifiers* I *and a function* $m : E \to I$ *such that*

$$\forall x \in C, \forall e_1, e_2 \in x, m(e_1) \neq m(e_2). \qquad \text{(Collision Freeness)}$$

We call $\mathcal{C}$ the underlying configuration structure *of* $\mathcal{I}$*, write* $\mathcal{F}(\mathcal{I}) = \mathcal{C}$*,* $\mathcal{I} = \mathcal{C} \oplus m$*, (*$I$ *being the domain of* m*), and write* $\mathbf{0}$ *for the identified configuration structure with* $E = \emptyset$*.*

For the rest of this paper, we assume always given $\mathcal{I} = (E, C, L, \ell, I, m)$ and $\mathcal{I}_i = (E_i, C_i, L_i, \ell_i, I_i, m_i)$, for $i = 1, 2$. Using Definition 5, we define the following operations:

▶ **Definition 25** (Operations on $\mathcal{I}$-structures)**.**

The product *of* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *is* $\mathcal{I}_1 \times \mathcal{I}_2 = (E, C, L, \ell, I, m)$*:*

- $(E_1, C_1, L_1, \ell_1) \times (E_2, C_2, L_2, \ell_2) = (E, C, L, \ell)$ *is the product of configuration structures with projections* $\pi_i : (E, C, L, \ell) \to (E_i, C_i, L_i, \ell_i)$*;*
- $I = (I_1 \cup \{m_\star\}) \times (I_2 \cup \{m_\star\})$*, for* $m_\star \notin I_1 \cup I_2$*;*
- $m : E_1 \times_\star E_2 \to I$ *is defined as*

$$m(e) = \begin{cases} (m_1(\pi_1(e)), m_\star) & \text{if } \pi_2(e) = \star \\ (m_\star, m_2(\pi_2(e))) & \text{if } \pi_1(e) = \star \\ (m_1(\pi_1(e)), m_2(\pi_2(e))) & \text{otherwise} \end{cases}$$

with the projections $\theta_i : I \to I_i \cup \{m_\star\}$ *and* $\theta_i(j_1, j_2) = j_i$*.*

Define the projections $\gamma_i : \mathcal{I}_1 \times \mathcal{I}_2 \to \mathcal{I}_i$ *as the pair* (π_i, θ_i)*.*

The relabeling *of* $\mathcal{I}_1$ *along* $r : E_1 \to L$ *is* $r \circ \mathcal{I}_1 = (E_1, C_1, L, r, I_1, m_1)$*.*

The restriction of a set of events $A \subseteq E_1$ *in* $\mathcal{I}_1$ *is* $\mathcal{I}_1 \restriction_A = (E_1, C_1, L_1, \ell_1) \restriction_A \oplus (m_1 \restriction_{E_1 \setminus A})$*.*

Similarly we define the restriction of a name.

The parallel composition *of $\mathcal{I}_1$ and $\mathcal{I}_2$ is $\mathcal{I}_1 \mid \mathcal{I}_2 = (E, C, L, \ell, I, m)$, with*

- *$\mathcal{I}_1 \times \mathcal{I}_2 = \mathcal{I}_3 = (E_3, C_3, L_3, \ell_3, I_3, m_3)$ is the product of $\mathcal{I}$-structures.*
- *$r \circ \mathcal{I}_3$, with $r : E_3 \to L \cup \{\bot\}$ defined as follows,*

$$r(e) = \begin{cases} \bot & \text{if } \pi_1(e) = e_1 \neq \star \wedge \pi_2(e) = e_2 \neq \star \wedge m_1(e_1) \neq m_2(e_2) \quad \text{(Err. 1)} \\ & \text{or if } \pi_1(e) = e_1 \neq \star \wedge \pi_2(e) = \star \wedge \exists e_2 \in E_2, m_1(e_1) = m_2(e_2) \quad \text{(Err. 2)} \\ & \text{or if } \pi_2(e) = e_2 \neq \star \wedge \pi_1(e) = \star \wedge \exists e_1 \in E_1, m_1(e_1) = m_2(e_2)) \quad \text{(Err. 3)} \\ \tau & \text{if } \pi_1(e) = e_1 \neq \star \wedge \pi_2(e) = e_2 \neq \star \wedge m_1(e_1) = m_2(e_2) \\ & \quad \wedge \ell_3(e) = (\alpha, \overline{\alpha}) \quad \text{(Sync.)} \\ \alpha & \text{if } \pi_1(e) = e_1 \neq \star \wedge \pi_2(e) = e_2 \neq \star \wedge m_1(e_1) = m_2(e_2) \\ & \quad \wedge \ell_3(e) = (\alpha, \alpha) \quad \text{(Fork)} \\ \ell_3(e) & \text{otherwise} \end{cases}$$

- *$(r \circ \mathcal{I}_3)\restriction_F = (E, C, L, \ell, I, m')$, where $F = \{e \in E_3 \mid r(e) = \bot\}$;*
- *m is defined as $m(e) = i$ for $m'(e) = (i, m_\star)$ or $m'(e) = (m_\star, i)$, or if $m'(e) = (i, j)$ and $i = j$.*

The postfixing *of the memory event $\langle i, \lambda, P\rangle$ to $\mathcal{I}_1$, is defined if i is not already an identifier in $\mathcal{I}_1$ as $\mathcal{I}_1 \cdot \langle i, \lambda, P\rangle = (E, C, L, \ell, I, m)$ where*

- *$E = E_1 \cup \{e\}$, for $e \notin E_1$,*
- *$C = C_1 \cup \{x \cup \{e\} \mid x \in C_1$ is maximal$\}$,*
- *$L = L_1 \cup \{\lambda\}$,*
- *$\ell = \ell_1 \cup \{e \mapsto \lambda\}$,*
- *$I = I_1 \cup \{i\}$,*
- *$m(e') = m_1(e')$ if $e' \neq e$, and $m(e) = i$ otherwise.*

In parallel compositions, r detects the wrong synchronization (or fork) pairs: if two events occur at the same time, then they must have the same identifier. In the case where an event occurs alone, then no other event can occur with the same identifier. The last step of the parallel composition consists in removing the spurious $m_\star$ from the events identifiers. Definition 27 will detail how those operations correspond to the encoding of a memory.

As detailed in Appendix A, $\mathcal{I}$-structures and "structure-preserving" functions form a category. This development supports the interest and validity of studying $\mathcal{I}$-structures, but can be omitted, except for the notion of isomorphism, that echoes Definition 6:

▶ **Definition 26.** *We write $\mathcal{I}_1 \cong \mathcal{I}_2$ if there exists an isomorphism $q = (f, f_m)$, i.e., an isomorphism $f = (f_E, f_L, f_C)$ between $\mathcal{F}(\mathcal{I}_1)$ and $\mathcal{F}(\mathcal{I}_2)$ endowed with a function $f_m : I_1 \to I_2$ that preserves identifiers: $f_m(m_1(e)) = m_2(f_E(e))$.*

4.2 Encoding the Memory of Reversible Processes

We show and illustrates in this section how $\mathcal{I}$-structures can encode memories, and highlight some properties of this encoding that will be useful in proving our main results.

▶ **Definition 27** (Encoding a RCCS memory)**.** *The encoding of the memory of a RCCS process in a $\mathcal{I}$-structure is defined by induction on the process and on the memory:*

$$\lceil m \rhd P \rceil = \lfloor m \rfloor \qquad \lceil R_1 \mid R_2 \rceil = \lceil R_1 \rceil \mid \lceil R_2 \rceil \qquad \lceil R \backslash a \rceil = \lceil R \rceil$$
$$\lfloor \langle i, \lambda, P\rangle . m \rfloor = \lfloor m \rfloor \cdot \langle i, \lambda, P\rangle \qquad \lfloor \Upsilon . m \rfloor = \lfloor m \rfloor \qquad \lfloor \emptyset \rfloor = \mathbf{0}$$

The memories of any RCCS process could be encoded into $\mathcal{I}$-structures, but we will consider only reachable processes (Definition 15), and thus coherent memories.

For the rest of this paper, we assume given reachable reversible processes R, R_1, R_2 and S, we write O_R for the origin of R, $\lceil R \rceil$ as $(E_R, C_R, \ell_R, I_R, m_R)$ and similarly for S.

The underlying configuration of $\lceil R \rceil$ is included in $[\![O_R]\!]$. We come back on this observation in Sect. 5, but we can observe that it is suggested by the following examples.

▶ **Example 28.** Looking back at the transition detailed in Example 14, we can observe that:

$$\begin{aligned}&\lceil(\langle 3,b,0\rangle.\langle 2,a,0\rangle.\ \Upsilon\ .\emptyset \rhd 0) \mid (\langle 2,\overline{a},0\rangle.\langle 1,c,0\rangle.\ \Upsilon\ .\emptyset \rhd 0)\rceil\\ =&\lceil\langle 3,b,0\rangle.\langle 2,a,0\rangle.\ \Upsilon\ .\emptyset \rhd 0\rceil \mid \lceil\langle 2,\overline{a},0\rangle.\langle 1,c,0\rangle.\ \Upsilon\ .\emptyset \rhd 0\rceil\\ =&\lfloor\langle 3,b,0\rangle.\langle 2,a,0\rangle.\ \Upsilon\ .\emptyset\rfloor \mid \lfloor\langle 2,\overline{a},0\rangle.\langle 1,c,0\rangle.\ \Upsilon\ .\emptyset\rfloor\end{aligned}$$

We can let $L = \{a, \overline{a}, b, c\}$, $\ell(x) = x$ for $x \in L$, $I = \{1, 2, 3\}$, and have:

$$\lfloor\langle 3,b,0\rangle.\langle 2,a,0\rangle.\ \Upsilon\ .\emptyset\rfloor = (\{a,b\}, \{\emptyset, \{a\}, \{a,b\}\}, L, \ell, I, \{a \mapsto 2, b \mapsto 3\})$$
$$\lfloor\langle 2,\overline{a},0\rangle.\langle 1,c,0\rangle.\ \Upsilon\ .\emptyset\rfloor = (\{c,\overline{a}\}, \{\emptyset, \{c\}, \{c,\overline{a}\}\}, L, \ell, I, \{c \mapsto 1, \overline{a} \mapsto 2\})$$

whose underlying structures are in Fig. 3a and 3b. The product of those two configurations gives the following set of events (where we keep naming the events after their label):

$(a,\star)$	(ev. 1)	$(\star,c)$	(ev. 3)	(a,c)	(ev. 5)	(b,c)	(ev. 7)
$(b,\star)$	(ev. 2)	$(\star,\overline{a})$	(ev. 4)	$(a,\overline{a})$	(ev. 6)	$(b,\overline{a})$	(ev. 8)

When doing the parallel composition, the relabeling step labels with $\perp$ events ev. 5, ev. 7 and ev. 8, since the event identifiers do not match in the pair, as well as ev. 1 and ev. 4, since they have the same identifier but occur unsynchronized. The relabeling also changes the labels of events ev. 2, ev. 3 and ev. 6 into b, c and τ, respectively. Hence, we obtain the $\mathcal{I}$-structure whose underlying structure is the gray part of Fig. 3c, with $m(\star, c) = m(c) = 1$, $m(a, \overline{a}) = m(\tau) = 2$ and $m(b, \star) = m(b) = 3$. Observe that the structure underlying the encoding of the memory is just a particular "path" in that configuration structure.

▶ **Example 29.** The encoding of the memory resulting from the (partial) execution

$$\emptyset \rhd (a.b \mid c) \xrightarrow{1:c} \xrightarrow{2:a} (\langle 2,a,0\rangle.\ \Upsilon\ .\emptyset \rhd b) \mid (\langle 1,c,0\rangle.\ \Upsilon\ .\emptyset \rhd 0)$$

has for underlying structure the gray part in Fig. 3d, with $m(c) = 1$ and $m(a) = 2$.

$\mathcal{I}$-structures resulting from the encoding of processes have interesting properties:

▶ **Lemma 30.** *For all $e_1, e_2 \in E_R$, $m_R(e_1) = m_R(e_2)$ implies $e_1 = e_2$.*

In particular, all memory events in R are either causally linked or concurrent, therefore the encoding of the memory of R results in a partially ordered set with one maximal element (Definition 2), linked by the subset relation. This was illustrated by the previous two examples and is proved by induction on the RCCS process.

▶ **Lemma 31.** *$\lceil R \rceil$ is a partially ordered set (poset) with one maximal configuration.*

4.3 (Hereditary) History-Preserving Bisimulations on CCS

Our main task now is to define relations on CCS, that we named HPB and HHPB (Definition 32), and then to prove that those relations correspond to their counterparts on the encoding of terms (Theorem 36) and that HHPB corresponds to B&F bisim on all processes (Corollary 38). This requires a notion of maximal event that relates to the operational semantics of RCCS (Lemma 34), and to prove that isomorphisms between $\mathcal{I}$-structures and label- and order-preserving bijections on the configurations are the same concept (Lemma 35).

▶ **Definition 32** (HPB and HHPB on RCCS). *A relation $\mathcal{R} \subseteq \mathsf{R} \times \mathsf{R} \times (E_1 \rightharpoonup E_2)$ such that $(\emptyset \rhd O_{R_1}, \emptyset \rhd O_R, \emptyset) \in \mathcal{R}$ and if $(R_1, R_2, f) \in \mathcal{R}$ then f is an isomorphism between $\lceil R_1 \rceil$ and $\lceil R_2 \rceil$ and (10) and (11) (resp. (10–13)) hold is called a* history-*(resp.* hereditary history-*)* preserving bisimulation between R_1 and R_2.

$$\forall S_1, R_1 \xrightarrow{i:\alpha} S_1 \Rightarrow \exists S_2, g, R_2 \xrightarrow{j:\alpha} S_2, g\restriction_{\lceil R_1 \rceil} = f, (S_1, S_2, g) \in \mathcal{R} \tag{10}$$

$$\forall S_2, R_2 \xrightarrow{i:\alpha} S_2 \Rightarrow \exists S_1, g, R_1 \xrightarrow{j:\alpha} S_1, g\restriction_{\lceil R_1 \rceil} = f, (S_1, S_2, g) \in \mathcal{R} \tag{11}$$

$$\forall S_1, R_1 \overset{i:\alpha}{\rightsquigarrow} S_1 \Rightarrow \exists S_2, f, R_2 \overset{j:\alpha}{\rightsquigarrow} S_2, g = f\restriction_{\lceil S_1 \rceil}, (S_1, S_2, g) \in \mathcal{R} \tag{12}$$

$$\forall S_2, R_2 \overset{i:\alpha}{\rightsquigarrow} S_2 \Rightarrow \exists S_1, g, R_1 \overset{j:\alpha}{\rightsquigarrow} S_1, g = f\restriction_{\lceil S_1 \rceil}, (S_1, S_2, g) \in \mathcal{R} \tag{13}$$

If there exists a (H)HPB *relation between R_1 and R_2 (resp. between $\emptyset \rhd P_1$ and $\emptyset \rhd P_2$), we say that R_1 and R_2 (resp. P_1 and P_2)* are (H)HPB.

Note that the definitions above reflect Definition 18: the condition $(\emptyset \rhd O_{R_1}, \emptyset \rhd O_R, \emptyset) \in \mathcal{R}$ is intuitively the counterpart to the condition that $(\emptyset, \emptyset, \emptyset)$ has to be included in the relation on configuration structures. Also, f shares similarity with the label- and order-preserving bijection (this will be made formal in Lemma 35).

▶ **Definition 33** (Maximal event). *An event e is* maximal in $\mathcal{I}$ *if there is no event e' such that $e <_{x^m} e'$, for x^m a maximal configuration of $\mathcal{I}$.*

Note that by Lemma 31, the encoding of a memory has at most one maximal configuration, but that does not prevent it from having multiple maximal events: Fig. 3d is an $\mathcal{I}$- structure resulting from the encoding of a memory with two maximal events, labeled a and c.

The following lemma shows the operational correspondence between a RCCS process and its encoding in $\mathcal{I}$-configuration structure.

▶ **Lemma 34.**

1. *For all transitions $R \xrightarrow{i:\alpha} S$, $\lceil R \rceil \cong \lceil S \rceil\restriction_{\{e\}}$ with e maximal in $\lceil S \rceil$ and $m_S(e) = i$.*
2. *For all transitions $R \overset{i:\alpha}{\rightsquigarrow} S$, $\lceil S \rceil \cong \lceil R \rceil\restriction_{\{e\}}$ with e maximal in $\lceil R \rceil$ and $m_R(e) = i$.*
3. *For any maximal event e in $\lceil R \rceil$, there is a transition $R \overset{m_R(e):\ell_R(e)}{\rightsquigarrow} S$ with $\lceil S \rceil \cong \lceil R \rceil\restriction_{\{e\}}$.*

The proof, for 1, shows that the transition triggered the creation of a maximal event with the same identifier, and nothing else, and that this event can be "traced" in $\lceil S \rceil$. It uses intermediate lemma showing how maximal events are preserved by certain operations on $\mathcal{I}$-structures. Part 2 follows easily, but 3 is more involved: it requires to show that e can be mapped to a particular transition in the trace from O_R to R, and, using a notion of trace equivalence, that this particular transition can be "postponed" and done last, so that R can backtrack on it.

▶ **Lemma 35.** *Letting x_1^m and x_2^m be the unique maximal configuration in $\lceil R_1 \rceil$ and $\lceil R_2 \rceil$, $\lceil R_1 \rceil \cong \lceil R_2 \rceil$ iff there exists a label- and order-preserving bijection between x_1^m and x_2^m.*

▶ **Theorem 36.** *P_1 and P_2 are* HHPB *(resp.* HPB*) iff $[\![P_1]\!]$ and $[\![P_2]\!]$ are.*

In the definition of (H)HPB on configuration structures (Definition 18), removing the condition that the bijection f on events must be preserved from one step to the next gives the definition of *weak*-HPB and *weak*-HHPB [3, 12]. The same loosening can be done on the definition of (H)HPB on RCCS (Definition 32), and a theorem similar to Theorem 36 regarding those "weak" relations can easily be proven.

▶ **Theorem 37.** HHPB *and* B&F bisim *coincide on all CCS processes.*

The proofs of those two results are relatively easy, once we have all the intermediate lemmas, and use one relation to define the other. Theorem 36 (resp. Theorem 37) uses our operational correspondence between RCCS processes (resp. RCCS memories) and their encodings as configuration structures [1, Lemma 6] (resp. as $\mathcal{I}$-structure (Lemma 34)) to transition between the semantic and syntactic worlds.

▶ **Corollary 38** (Main result). *P_1 and P_2 are* B&F bisim *iff $[\![P_1]\!]$ and $[\![P_2]\!]$ are* HHPB.

Proof. This is an immediate corollary of Theorem 37 and Theorem 36. ◀

5 Concluding Remarks

As we stressed, this work offers a "definitive" answer to the question of finding a relation for process algebras that corresponds to (H)HPB on their encodings for *all* processes. It uses the memory mechanism of RCCS, or, more precisely, the encoding of this memory into an "enriched" configuration structure, called identified configuration structure. This observation echoes our previous formalism [1] in an interesting way: as mentioned in Sect. 1, we offered to encode a reversible process R as a pair $([\![O_R]\!], x_R)$ made of the configuration structure encoding the origin of R, and a configuration x_R in it, called the address of R. The intuition was that we could "match" a partially executed process with a configuration. We can go further, and observe that the encoding of the memory of R has the same underlying configuration structure as the structure generated by its address, $x_R\downarrow$ (Definition 3). This lemma is actually used to prove Theorem 36, and it reads:

▶ **Lemma 39.** *The underlying configuration structure of $\lceil R \rceil$ is isomorphic to $x_R\downarrow$.*

We can now come back to the intuitions of Fig. 1 using Example 29: the encoding of the memory of $(\langle 2, a, 0\rangle.\Upsilon.\emptyset \rhd b) \mid (\langle 1, c, 0\rangle.\Upsilon.\emptyset \rhd 0)$ corresponds the "past" of the process, whose underlying structure is grayed out in Fig. 3d. What is left to execute, $b \mid 0$, corresponds to the "future" of that process, and is represented by the configuration $\{c, a, b\}$ in Fig. 3d. Note that the configuration $\{a, b\}$ is not part of the past nor of the future of this process, as reaching this configuration would require to undo c first.

Different versions of the back-and-forth bisimulation have been previously proposed for reversible processes and with the adequate restrictions on the class of processes, these versions coincide. But to capture HHPB on all process, B&F bisim uses both the reversibility and the memory mechanisms of RCCS. As we show with Example 23, using only the backward transitions does not capture HHPB on processes with auto-concurrency.

As future work, we plan to investigate the versions of HPB and HHPB where the τ transitions are ignored. We would also like to identify a criteria to determine what the "right" structural equivalence for RCCS is: as of now, our equivalence excludes e.g. $R \mid 0 \equiv R$ for technical reasons that should be overcome.

References

1 Clément Aubert and Ioana Cristescu. Contextual equivalences in configuration structures and reversibility. *J. Log. Algebr. Methods Program.*, 86(1):77–106, 2017. doi:10.1016/j.jlamp.2016.08.004.

2 Paolo Baldan and Silvia Crafa. Hereditary history-preserving bisimilarity: Logics and automata. In Jacques Garrigue, editor, *APLAS*, volume 8858 of *LNCS*, pages 469–488. Springer, 2014. doi:10.1007/978-3-319-12736-1_25.

3 Marek A. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Instytut Podstaw Informatyki PAN filia w Gdańsku, 1991. URL: `http://www.ipipan.gda.pl/~marek/papers/historie.ps.gz`.

4 Gérard Boudol and Ilaria Castellani. Permutation of transitions: An event structure semantics for CCS and SCCS. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop, Noordwijkerhout, The Netherlands, May 30 - June 3, 1988, Proceedings*, volume 354 of *LNCS*, pages 411–427. Springer, 1988. `doi:10.1007/BFb0013028`.

5 Vincent Danos and Jean Krivine. Reversible communicating systems. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004. `doi:10.1007/978-3-540-28644-8_19`.

6 Vincent Danos and Jean Krivine. Transactions in RCCS. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, volume 3653 of *LNCS*, pages 398–412. Springer, 2005. `doi:10.1007/11539452_31`.

7 Philippe Darondeau and Pierpaolo Degano. Causal trees: Interleaving + causality. In Irène Guessarian, editor, *Semantics of Systems of Concurrent Processes, LITP Spring School on Theoretical Computer Science, La Roche Posay, France, April 23-27, 1990, Proceedings*, volume 469 of *LNCS*, pages 239–255. Springer, 1990. `doi:10.1007/3-540-53479-2_10`.

8 Sibylle B. Fröschle and Thomas T. Hildebrandt. On plain and hereditary history-preserving bisimulation. In Miroslaw Kutylowski, Leszek Pacholski, and Tomasz Wierzbicki, editors, *MFCS*, volume 1672 of *LNCS*, pages 354–365. Springer, 1999. `doi:10.1007/3-540-48340-3_32`.

9 Iain Phillips and Irek Ulidowski. Reversing algebraic process calculi. In Luca Aceto and Anna Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *LNCS*, pages 246–260. Springer, 2006. `doi:10.1007/11690634_17`.

10 Iain Phillips and Irek Ulidowski. Reversibility and models for concurrency. *ENTCS*, 192(1):93–108, 2007. `doi:10.1016/j.entcs.2007.08.018`.

11 Iain Phillips and Irek Ulidowski. Reversing algebraic process calculi. *J. Log. Algebr. Program.*, 73(1-2):70–96, 2007. `doi:10.1016/j.jlap.2006.11.002`.

12 Iain Phillips and Irek Ulidowski. A logic with reverse modalities for history-preserving bisimulations. In Bas Luttik and Frank Valencia, editors, *EXPRESS*, volume 64 of *EPTCS*, pages 104–118, 2011. `doi:10.4204/EPTCS.64.8`.

13 Alexander Rabinovich and Boris Avraamovich Trakhtenbrot. Behavior structures and nets. *Fund. Inform.*, 11(4):357–404, 1988.

14 Vladimiro Sassone, Mogens Nielsen, and Glynn Winskel. Models for concurrency: Towards a classification. *Theoret. Comput. Sci.*, 170(1-2):297–348, 1996. `doi:10.1016/S0304-3975(96)80710-9`.

15 Robert J. van Glabbeek. History preserving process graphs. Technical report, Stanford University, 1996. URL: `http://kilby.stanford.edu/~rvg/pub/history.draft.dvi`.

16 Robert J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Inform.*, 37(4/5):229–327, 2001. `doi:10.1007/s002360000041`.

17 Glynn Winskel. Event structure semantics for CCS and related languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of *LNCS*, pages 561–576. Springer, 1982. `doi:10.1007/BFb0012800`.

18 Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, volume 255 of *LNCS*, pages 325–392. Springer, 1986. `doi:10.1007/3-540-17906-2_31`.

19 Glynn Winskel and Mogens Nielsen. Models for concurrency. In Samson Abramsky, Dov M. Gabbay, and Thomas Stephen Edward Maibaum, editors, *Semantic Modelling*, volume 4 of *Handbook of Logic in Computer Science*, pages 1–148. Oxford University Press, 1995.

A Category Theory

Configuration structures often use the insights provided by the categorical framework [1, 14, 17]. We regroup in this appendix the categorical treatment related to the configuration structures and to the identified configuration structures.

▶ **Definition 40** (Category of configuration structures). *We define $\mathbb{C}$ the category of configuration structures, where an object is a configuration structure, and a morphism $f : \mathcal{C}_1 \to \mathcal{C}_2$ is a triple (f_E, f_L, f_C) such that*

- *$f_E : E_1 \to E_2$ preserves labels: $\ell_2(f_E(e)) = f_L(\ell_1(e))$, for $f_L : L_1 \to L_2$;*
- *$f_C : C_1 \to C_2$ is defined as $f_C(x) = \{f_E(e) : e \in x\}$.*

If there exists an isomorphism $f : \mathcal{C}_1 \to \mathcal{C}_2$, then we write $\mathcal{C}_1 \cong \mathcal{C}_2$.

Observe that $\mathbb{C}$ has $\mathbf{0}$ for initial object. For simplicity, we often assume that $L_1 = L_2$, i.e., that all the configuration structures use the same set of labels, take f_L to be the identity and remove it from the notation.

▶ **Definition 41** (Category of $\mathcal{I}$-structures). *We define $\mathbb{D}$ the category of identified configuration structures, where objects are $\mathcal{I}$-structures, and a morphism $f : \mathcal{I}_1 \to \mathcal{I}_2$ is a tuple $q = (f, f_m)$ such that*

- *$f = (f_E, f_C)$ is a morphism in $\mathbb{C}$ between the underlying structures of $\mathcal{I}_1$ and $\mathcal{I}_2$,*
- *$f_m : I_1 \to I_2$ preserves identifiers: $f_m(m_1(e)) = m_2(f_E(e))$.*

If there exists an isomorphism $q : \mathcal{I}_1 \to \mathcal{I}_2$, then we write $\mathcal{I}_1 \cong \mathcal{I}_2$.

Observe that $\mathbb{D}$ as for initial object $\mathbf{0}$, and that $\mathbb{C}$ is a subcategory of $\mathbb{D}$. In both $\mathbb{C}$ and $\mathbb{D}$, composition is written $\circ$ and defined componentwise.

▶ **Lemma 42.** *Identified configuration structures and their morphisms form a category.*

Proof. Identity For every $\mathcal{I}$-structure $\mathcal{I} = (E, C, \ell, I, m)$, $\mathrm{id}_{\mathcal{I}} : \mathcal{I} \to \mathcal{I}$ is defined to be the identity on the underlying configuration structure $\mathrm{id} : (E, C, \ell) \to (E, C, \ell)$ from $\mathbb{C}$, that trivially preserves identifiers. For any morphism $f : \mathcal{I}_1 \to \mathcal{I}_2$, $f \circ \mathrm{id}_{\mathcal{I}_1} = f = \mathrm{id}_{\mathcal{I}_2} \circ f$ is trivial.

Associativity for $f : \mathcal{I}_1 \to \mathcal{I}_2$, $g : \mathcal{I}_2 \to \mathcal{I}_3$ and $h : \mathcal{I}_3 \to \mathcal{I}_4$, $h \circ (g \circ f) = (h \circ g) \circ f$ is inherited from the associativity in $\mathbb{C}$, and since f, g and h all preserves identifiers.

Hence $\mathbb{D}$ is a category. ◀

Unsurprisingly, a forgetful functor and an enrichment functor can be defined between those two categories. The only assumption is that we need to suppose that every configuration structure can be endowed with a total ordering $\preceq$ on its events.

▶ **Lemma 43.** *The forgetful functor $\mathcal{F} : \mathbb{D} \to \mathbb{C}$, defined by*

- *$\mathcal{F}(E, C, \ell, I, m) = (E, C, \ell)$*
- *$\mathcal{F}(f_E, f_C, f_m) = (f_E, f_C)$*

and $\mathcal{S} : \mathbb{C} \to \mathbb{D}$, defined by

- *$\mathcal{S}(E, C, \ell) = (E, C, \ell, I, m)$, where $I = \{1, \ldots, |E|\}$ for $|E|$ the cardinality of E, and*

$$m(e) = \begin{cases} 1 & \text{if } \forall e', e \preceq e' \\ i+1 & \text{if } \exists e', e' \preceq e,\ m(e') = i \text{ and there is no } e'' \text{ s.t. } e' \preceq e'' \preceq e \end{cases}$$

- *For $(f_E, f_C) : (E_1, C_1, \ell_1) \to (E_2, C_2, \ell_2)$, $\mathcal{S}(f_E, f_C) = (f_E, f_C, f_m)$, where we let $f_m(m_1(e)) = m_2(f_E(e_2))$.*

are functors.

Proof. Proving that $\mathcal{F}$ is a functor is immediate.

Proving that $\mathcal{S}(\mathcal{C})$ is a $\mathcal{I}$-structure is immediate, since our construction of m trivially insures Collision Freeness. For $(f_E, f_C) : \mathcal{C}_1 \to \mathcal{C}_2$, proving that $\mathcal{S}(f_E, f_C)$ is a morphism between $\mathcal{S}(\mathcal{C}_1)$ and $\mathcal{S}(\mathcal{C}_2)$ is also immediate. For the preservation of the identity, we compute:

$$\begin{aligned}\mathcal{S}(\mathrm{id}_{\mathcal{C}}) &= \mathcal{S}(\mathrm{id}_E, \mathrm{id}_C)\\ &= (\mathrm{id}_E, \mathrm{id}_C, f_m)\end{aligned}$$

where $f_m(m(e)) = m(\mathrm{id}_E(e)) = m(e)$, hence $f_m = \mathrm{id}_I : I \to I$,

$$\begin{aligned}&= (\mathrm{id}_E, \mathrm{id}_C, \mathrm{id}_{\mathsf{I}})\\ &= \mathrm{id}_{\mathcal{S}(\mathcal{C})}\end{aligned}$$

For the composition of morphisms, given $f = (f_C, f_E) : \mathcal{C}_1 \to \mathcal{C}_2$ and $g = (g_C, g_E) : \mathcal{C}_2 \to \mathcal{C}_3$, we write $\mathcal{S}(\mathcal{C}_i) = (E_i, C_i, \ell_i, I_i, m_i)$ and we compute:

$$\begin{aligned}\mathcal{S}(g) \circ \mathcal{S}(f) &= (g_C, g_E, g_m) \circ (f_C, f_E, f_m)\\ &= (g_C \circ f_C, g_E \circ f_E, g_m \circ f_m)\end{aligned}$$

where, for all $e \in E_1$, we compute:

$$\begin{aligned}(g_m \circ f_m)(m_1(e)) &= g_m(f_m(m_1(e))\\ &= g_m(m_2(f_E(e))) && \text{(Since } f_m \text{ preserves identifiers)}\\ &= m_3(g_E(f_E(e))) && \text{(Since } g_m \text{ preserves identifiers)}\end{aligned}$$

Hence we can conclude:

$$\mathcal{S}(g) \circ \mathcal{S}(f) = \mathcal{S}(g \circ f)$$ ◀

▶ Remark 44. In $\mathbb{D}$, every morphism $f = (f_E, f_L, f_C, f_m)$ from $\mathcal{I}_1$ to $\mathcal{I}_2$ is actually fully determined by f_E whenever $f_L = \mathrm{id}$. Indeed, given $f_E : E_1 \to E_2$, then we can define for all $x \in C_1$, $f_C(x) = \{f_E(e) \mid e \in x\}$ and f_m as $f_m(m_1(e)) = m_2(f_E(e))$. We will often make the abuse of notation of writing f_E for f and reciprocally.

▶ **Lemma 45.** *The product of $\mathcal{I}$-structures is the product in $\mathbb{D}$.*

Proof. First note that $\mathcal{I}_1 \times \mathcal{I}_2$ is an $\mathcal{I}$-structure as it is a configuration structure endowed with an m function, where by definition every event in a configuration has a unique label. Secondly, it is easy to show that the projections $\gamma_i : \mathcal{I}_1 \times \mathcal{I}_2 \to \mathcal{I}_i$ are morphisms, since they are defined as the pair (π_i, θ_i). Lastly to show that the structure $\mathcal{I}_1 \times \mathcal{I}_2$ has the universal property, we proceed in two steps:

- the underlying configuration structure is the product of the underlying configuration structures, by definition of the product in configuration structure:

  $$\mathcal{F}(\mathcal{I}_1 \times \mathcal{I}_2) = \mathcal{F}(\mathcal{I}_1) \times \mathcal{F}(\mathcal{I}_2);$$

- for any $\mathcal{I}'$ which projects into $\mathcal{I}_1$ and $\mathcal{I}_2$, then $\mathcal{F}(\mathcal{I}')$ projects into $\mathcal{F}(\mathcal{I}_1)$ and $\mathcal{F}(\mathcal{I}_2)$ and therefore there exists a unique morphism $h : \mathcal{F}(\mathcal{I}') \to \mathcal{F}(\mathcal{I}_1 \times \mathcal{I}_2)$. It is easy to show that since the projections preserve identifiers, then so does h which concludes our proof.

This lemma also follows from [19, Proposition 85]. ◀

B Proofs and Auxiliary Lemmas

In this section we detail the proofs missing from the main text, as well as introducing some intermediate results that are necessary for the proofs.

B.1 Proof for Sect. 3.2

▶ **Theorem 22.** *On the class of processes without auto-concurrency,* B&F bisim *and* SB&F bisim *are the same relations.*

Proof. Let R_1, R_2 be two reversible processes without auto-concurrency. We want to show that

1. $(R_1, R_2) \in \mathcal{R}$ for $\mathcal{R}$ a SB&F bisim implies that there exists f a bijection between $\mathsf{I}(R_1)$ and $\mathsf{I}(R_2)$ such that $(R_1, R_2, f) \in \mathcal{R}'$ is a B&F bisim.
2. $(R_1, R_2, f) \in \mathcal{R}'$ for $\mathcal{R}'$ a B&F bisim implies that $(R_1, R_2) \in \mathcal{R}$ for $\mathcal{R}$ a SB&F bisim.

For 1, we first note that there exists forward-only traces from $\emptyset \rhd O_{R_1}$ to R_1 [5, Lemma 10]. We pick one, θ_1, and construct the bijection f between $\mathsf{I}(R_1)$ and $\mathsf{I}(R_2)$ using it. We start by letting $(\emptyset \rhd O_{R_1}, \emptyset \rhd O_{R_2}, \emptyset) \in \mathcal{R}'$. Then, assuming θ_1 starts with $\emptyset \rhd O_{R_1} \xrightarrow{i:\alpha} R_1'$, since $(R_1, R_2) \in \mathcal{R}$, we know there are R_2' and j such that $\emptyset \rhd O_{R_2} \xrightarrow{j:\alpha} R_2'$ and $(R_1', R_2') \in \mathcal{R}$, and we let $(R_1', R_2', \{i \to j\}) \in \mathcal{R}'$. We iterate this construction until we obtain the bijection f such $(R_1, R_2, f) \in \mathcal{R}'$. That this bijection can be extended in case of forward transitions from R_1 or R_2 is obvious, using $\mathcal{R}$. The more difficult part of the proof is to show that f is "right", i.e., that if R_1 or R_2 does a backward transition to a term S, and if S is paired with S' in $\mathcal{R}$, then it is the case that the identifiers of the transitions leading to S and S' are "matched", i.e., in bijection in f.

Stated formally, we want to show that for any $(R_1, R_2) \in \mathcal{R}$ with f constructed as above, then for all $R_1 \overset{i:\alpha}{\rightsquigarrow} S_1$ and $R_2 \overset{j:\alpha}{\rightsquigarrow} S_2$ such that $(S_1, S_2) \in \mathcal{R}$, $\{i \to j\} \in f$. We show it by contradiction: let us then take $(R_1, R_2) \in \mathcal{R}$ and $f : \mathsf{I}(R_1) \to \mathsf{I}(R_2)$. We want to show that $(R_1, R_2, f) \in \mathcal{R}'$ is a valid relation, and for this we must show that it can accommodate the backward transitions. Suppose that we have two pairs of indexes $\{i \to j\}, \{i' \to j'\}$ in f, and that $R_1 \overset{i':\alpha}{\rightsquigarrow} S_1$ and that $R_2 \overset{j:\alpha}{\rightsquigarrow} S_2$ such that $(S_1, S_2) \in \mathcal{R}$.

First, observe that all four indexes, i, i', j and j', are associated to the same label α: due to the way f was constructed, following $\mathcal{R}$, it cannot be the case that two transitions with different labels have their identifiers paired. Secondly, since $R_2 \overset{j:\alpha}{\rightsquigarrow} S_2$ and $\{i \to j\}$ is in f, it must be the case that there is a transition in θ_1 with the identifier i by construction of f. Let us denote $t_i : S_1' \xrightarrow{i:\alpha} R_1'$ and $t_{i'} : S_1 \xrightarrow{i':\alpha} R_1$.

The following paragraphs are necessary for the proof but are more general and can also be read as standalone.

Interlude: Concurrency in a Trace and Trace Equivalence Concurrency on events corresponds to a notion of concurrency on transitions in a trace. We introduce here the concepts needed for this proof, as they are defined in [5, Definition 7 and Lemma 8].

Two transitions $t_1 = R \xrightarrow{i_1:\alpha_1} R_1$ and $t_2 = R' \xrightarrow{i_2:\alpha_2} R_2$ are *composable* if $R_1 = R'$, and doing t_1, then t_2 is written as the composition $t_1; t_2$. Given n composable transitions $t_i : R_i \xrightarrow{i:\alpha_i} R_{i+1}$ and their composition $t_1; \ldots; t_n$, we say that t_i *is a direct cause of* t_k for $1 \leqslant i < k \leqslant n$ and write $t_i < t_k$ (or, for short, $i < k$) if there is a memory stack m in R_{i+1} and a memory stack m' in R_{k+1} such that $m < m'$, where the order on memory stacks is given by prefix ordering.

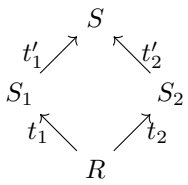
Let $R \xrightarrow{i:\alpha} S$ be a transition. If $\alpha \neq \tau$, we write $m_{R/S} = \{m\}$ where

$$R = (\cdots((R_3 \mid ((R_1 \mid (m \rhd P)) \mid R_2)\backslash\overrightarrow{b^1}) \mid R_4)\backslash\overrightarrow{b^2} \cdots \mid R_n)\backslash\overrightarrow{b^m}$$
$$S = (\cdots((R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash\overrightarrow{b^1}) \mid R_4)\backslash\overrightarrow{b^2} \cdots \mid R_n)\backslash\overrightarrow{b^m}$$

for some R_i any of which could be missing and for some $\overrightarrow{b^j}$, possibly missing as well. If $\alpha = \tau$, then $m_{R/S}$ will contain the pair of memory stacks that has been changed by the transition. Intuitively, the notation $m_{R/S}$ is useful below to extract the memory stack(s) modified by a forward transition from R to S.

Two transitions are *coinitial* if they have the same source process and *cofinal* if they have the same target process. We say that two coinitial transitions $t_1 = R \xrightarrow{i_1:\alpha_1} S_1$ and $t_2 = R \xrightarrow{i_2:\alpha_2} S_2$ are *concurrent* if $m_{R/S_1} \cap m_{R/S_2} = \emptyset$, that is, if the transitions modify disjoint memories in R.

The square lemma [5, Lemma 8] says that moreover, given two such concurrent transitions, there exists two cofinal and concurrent transitions $t'_1 = S_1 \xrightarrow{i_2:\alpha_2} S$ and $t'_2 = S_2 \xrightarrow{i_1:\alpha_1} S$. The name of the lemma comes from this picture:



Moreover, the traces $\theta_1 = t_1; t'_1$ and $\theta_2 = t_2; t'_2$ are *equivalent* [5, Definition 9]. This allows one to define *equivalence classes on transitions*: t_1 in θ_1 is equivalent to t'_2 in θ_2 if θ_1 is equivalent to θ_2 and t_1 and t'_2 have the same index. Then in the trace $t_1; t'_1$ we are now allowed to say that t_1 is concurrent to t'_1.

In a trace $t_1; t_2$ we have that t_1 is concurrent to t_2 iff t_1 is not a cause of t_2. This follows from a case analysis using the definitions of concurrency and causality. Thanks to trace equivalence, we also have that in a trace $t_1; \ldots; t_n$ either t_1 is a cause of t_n or the two transitions are concurrent. We refer the reader to [4] for the complete treatment of concurrency and causality in the trace of a CCS process.

The definitions of concurrency for forward coinitial traces and of causality for forward traces can easily be "flipped" into definitions of concurrency for backward cofinal traces, and of causality for backward traces. Also, the concurrency and causality relations between transitions can be "lifted" to one between indexes in a trace. We will use them below.

Back to our Proof Observe that i and i' cannot be concurrent in θ_1. If they were, then in the encoding of R_1, there would be two events–the one introduced by t'_i and the one introduced by t_j–that are concurrent, with the same label, but that is not possible since R_1 is without auto-concurrency. Since two transitions are either concurrent or causal, it follows that either $i < i'$ or $i' < i$.

- As the cause cannot backtrack before the effect it follows that the case $i' < i$ is not possible.
- Then $i < i'$. We now reason by cases on $j < j'$ or $j' < j$.
  - If $j < j'$ then R_1 must have already backtracked on j', which implies that the there is no j' in f as we assumed.

- The case $j' < j$ is not possible. Note that $i < i'$ implies an order in which the pair of indexes are added to the bijection, in this case the pair $\{i \to j\}$ occurs before $\{i' \to j'\}$ and therefore it cannot be that $j' < j$.

From this reasoning, we get that for any $R_1 \overset{j:\alpha}{\rightsquigarrow} S_1$ and $R_2 \overset{j:\alpha}{\rightsquigarrow} S_2$ such that $(S_1, S_2) \in \mathcal{R}$, $\{i \to j\} \in f$. We still need to show that $(S_1, S_2, f \setminus \{i \to j\}) \in \mathcal{R}'$ is a "valid" relation. We only need to show that $f \setminus \{i \to j\}$ matches the "right" indices in case of a backward transition, but we can simply iterate the previous reasoning until we reach $(\emptyset \rhd O_{R_1}, \emptyset \rhd O_{R_2}, \emptyset) \in \mathcal{R}'$. This concludes this direction of the proof.

For 2, there is nothing to prove, since we are dropping a requirement, the existence of the bijection. ◀

B.2 Operations on Identified Configurations Preserve Them (Sect. 4.1)

Our main goal here is to prove that the operations of Definition 25 preserve $\mathcal{I}$-structures (Lemma 48), and that requires the following intermediate lemma:

▶ **Lemma 46.** *For every reversible thread $m \rhd P$ of a reachable process R, and for all $i \in \mathsf{I}(m)$, i occurs once in m.*

Proof. We prove it by induction on the structure of m.

- if $m = \emptyset$, then it is obvious.
- if $m = \Upsilon.m'$, then by induction hypothesis, for all $i \in \mathsf{I}(m')$, i occurs only once in m', and since no identifier occur in Υ, i occurs only once in m.
- if $m = \langle j, \lambda, Q\rangle.m'$ then there exists S such that $m' \rhd S \xrightarrow{j:\lambda} \langle j, \lambda, Q\rangle.m' \rhd P$ since R is reachable. By induction we know that for all $i \in \mathsf{I}(m')$, i occurs once in m'. We reason on the derivation tree of the transition that add the memory event $\langle j, \lambda, Q\rangle$ and we have that for any such transition, the rule act. of Fig. 4 is applied as axiom at the top of the derivation tree. By the side condition of the rule act., we know that $j \notin \mathsf{I}(m')$, hence that for all $i \in \mathsf{I}(m)$, i occurs once in m. ◀

▶ Remark 47. Note that the property above holds for reversible threads, and not for RCCS processes in general: we actually *want* memory events to sometimes share the same identifiers. Indeed, two memory events need to have the same identifiers if they result from a synchronization (i.e., the application of the syn. rule of Fig. 4) or a fork (i.e., the application of the Distribution of Memory rule of structural equivalence, Definition 13).

▶ **Lemma 48.** *The operations of Definition 25 (product, relabeling, restriction, parallel composition and postfixing) preserve $\mathcal{I}$-structures.*

Proof. Let us note that (i) the product, relabeling, restriction, and parallel composition on configuration structures from Definition 5 preserve configuration structures, as their are adaptations from [17, 18], and that (ii) any configuration structure endowed with a valid identifier function (i.e., such that no two events in the same configuration have the same identifier, cf. Collision Freeness) is a valid $\mathcal{I}$-structure.

For the product, it follows trivially from Lemma 45.

Relabeling does not change anything but the labels, so we have nothing to prove.

The restriction only removes events in configurations and keeps the identifier function intact. Hence if the initial structure has a valid identifier function, then the identifier function of the new structure is a valid one by assumption.

Let us now consider the parallel composition of two $\mathcal{I}$-structures, denoted $\mathcal{I}_1 \mid \mathcal{I}_2 = (E, C, L, \ell, I, m)$. Proving that the identifier function m is valid follows from a case analysis. Given a configuration $x \in C$ and two events $e, e' \in x$, these are the possible cases:

- $\pi_2(e) = \star$ and $\pi_2(e') = \star$. In this case, looking at the definition of the product in $\mathcal{I}$-structures, $m(e) = (m_1(\pi_1(e)), m_\star)$ and $m(e') = (m_1(\pi_1(e')), m_\star)$. If $m(e) = m(e')$, then $m_1(\pi_1(e)) = m_1(\pi_1(e'))$ in the configuration $\pi_1(x)$ in $\mathcal{I}_1$. But that's a contradiction, since $\pi_1(e)$ and $\pi_1(e')$ are in the same configuration and the identifier function of $\mathcal{I}_1$ is valid.
- $\pi_1(e) = \star$ and $\pi_1(e') = \star$. This case is similar as the previous one, except that it uses that the identifier function of $\mathcal{I}_2$ is valid.
- $\pi_1(e) \neq \star$ and $\pi_2(e') \neq \star$ (with either $\pi_1(e') = \star$ or $\pi_1(e') \neq \star$). If $m(e) = m(e')$, then $m_i(\pi_i(e)) = m_i(\pi_i(e'))$ for $i = 1, 2$. Then in this case,
  - either one of them, say, e, is a synchronization or a fork: in this case, $m_1(\pi_1(e)) = m_2(\pi_2(e)) = m_2(\pi_2(e'))$, and e' was relabeled $\bot$ at the relabeling stage of the parallel composition, and then removed during the restriction. Hence a contradiction: e and e' cannot be two events in the same configuration.
  - or none of them is a synchronization, in which case both events were removed by the restriction. Hence, again, a contradiction: e and e' cannot be two events in the same configuration.

  The symmetric case (where $\pi_2(e) \neq \star$ and $\pi_1(e') \neq \star$) is similar.

For the postfixing operation, we have to first prove that the underlying configuration of $\mathcal{I}_1 \cdot \langle i, \lambda, P\rangle$, written (E, C, L, ℓ), is a valid configuration structure. Finiteness is satisfied because the underlying configuration of $\mathcal{I}_1$, written (E_1, C_1, L_1, ℓ_1), is a configuration structure, and every configuration in $\{x \cup \{e\} \mid x \in C_1 \text{ is maximal}\}$ is finite. For Coincidence Freeness, we also only have to check the configurations in $\{x \cup \{e\} \mid x \in C_1 \text{ is maximal}\}$. Given $y \in \{x \cup \{e\} \mid x \in C_1 \text{ is maximal}\}$, there exists y' such that $y = y' \cup \{e\}$. Given two events e_1, e_2 in y such that $e_1 \neq e_2$, if $e_1 = e$ or $e_2 = e$, then y' is the configuration we are looking for. Otherwise, it follows from y' being a configuration in the configuration structure (E_1, C_1, L_1, ℓ_1). Finite Completeness and Stability trivially follow from the fact that (E_1, C_1, L_1, ℓ_1) is a configuration structure. Hence, $\mathcal{I}_1 \cdot \langle i, \lambda, P\rangle$ has a configuration structure underlying, and we just need to prove that the identifier function is valid. That follows from the fact that i is not an identifier in the codomain of m_i, and hence $\mathcal{I}_1 \cdot \langle i, \lambda, P\rangle$ is an $\mathcal{I}$- structure. ◀

B.3 Properties of Memory Encodings (Sect. 4.2)

Remember that we assume given coherent reversible processes R, R_1, R_2 and S (without any restriction), and that we write O_R for the origin of R, and $\lceil R \rceil$ as $(E_R, C_R, \ell_R, I_R, m_R)$ and similarly for S.

▶ **Lemma 30.** *For all $e_1, e_2 \in E_R$, $m_R(e_1) = m_R(e_2)$ implies $e_1 = e_2$.*

Proof. We proceed by structural induction on R. From Lemma 46 the only interesting case is the parallel composition, i.e. $R = R_1 \mid R_2$. From the definition of parallel composition in $\mathcal{I}$-structures (Definition 25), it follows that $m_R(e_1) = m_R(e_2)$ implies $e_1 = e_2$. ◀

▶ **Lemma 31.** *$\lceil R \rceil$ is a partially ordered set (poset) with one maximal configuration.*

Proof. We proceed by induction on R.

If R is $m \rhd P$, we prove that $\lceil m \rhd P \rceil$ is a poset with one maximal element by induction on m. The base case, if m is $\emptyset$, is trivial, since $\lfloor \emptyset \rfloor = \mathbf{0}$ is a poset with one maximal element, $\emptyset$. If m is $\Upsilon.m'$, then it follows by induction hypothesis, since $\lfloor \Upsilon.m' \rfloor = \lfloor m' \rfloor$. If m is $\langle i, \alpha, P\rangle.m'$, then by induction hypothesis, $\lfloor m' \rfloor$ is a poset with one maximal element, and

the postfixing construction used to define $\lfloor\langle i, \alpha, P\rangle.m'\rfloor$, detailed in Definition 25, preserves that property.

If R is $R'\backslash a$, then it trivially follows by induction hypothesis.

Finally, if R is $R_1 \mid R_2$, then by induction hypothesis we get that $\lceil R_1 \rceil$ and $\lceil R_2 \rceil$ are both posets with a maximal configuration. We also know by Lemma 30 that events in each structure have disjoint identifiers (however the two structures can share identifiers with each other). Looking at the definition of parallel composition for $\mathcal{I}$-structures (Definition 25), we may observe that $\lceil R \rceil = \lceil R_1 \mid R_2 \rceil = \lceil R_1 \rceil \mid \lceil R_2 \rceil$ consists of the structure $\big(r \circ (\lceil R_1 \rceil \times \lceil R_2 \rceil)\big)\restriction_F$ with a different m function.

We show that there exists more than one maximal configurations in $\lceil R_1 \rceil \times \lceil R_2 \rceil$ and that all but one are removed by the restriction.

We show this by first showing that there exists more than one maximal configurations in $\mathcal{F}(\lceil R_1 \rceil) \times \mathcal{F}(\lceil R_2 \rceil)$, denoted here with $\mathcal{C}$. From the definition of product (Definition 5), we have that there exists $y_1, \ldots, y_n$ maximal configurations in $\mathcal{C}$ such that $\pi_1(y_i)$ and $\pi_2(y_i)$ are maximal in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. As $\lceil R_1 \rceil \times \lceil R_2 \rceil = (\mathcal{F}(\lceil R_1 \rceil) \times \mathcal{F}(\lceil R_2 \rceil)) \oplus m$ it follows that the maximal configurations of $\mathcal{C}$ are preserved in $\lceil R_1 \rceil \times \lceil R_2 \rceil$.

A second step is then to show that the restriction keeps only one maximal configuration. Let y_i, y_j be two maximal configurations. As they are maximal it implies that $y_i \cup y_j \notin \mathcal{C}$, for $i \neq j \leqslant n$. In turn, this implies that there exists $e_i \in y_i$ and $e_j \in y_j$ such that $\pi_1(e_i) = \pi_1(e_j)$ and $e_i \neq e_j$, as otherwise $y_i \cup y_j$ would be defined. Here we assume that $\pi_1(e_i) = \pi_1(e_j)$ but we could also take $\pi_2(e_i) = \pi_2(e_j)$ and the argument still holds.

Let us now take d an event in $\mathcal{C}_1$ and take $e_1, \ldots, e_m$ the subset of events in E where $\pi_1(e_i) = d$. The restriction in the parallel composition of $\lceil R_1 \rceil \mid \lceil R_2 \rceil$ keeps only one such event e_i and removes the rest. Therefore, from all maximal configurations $y_1, \ldots, y_m$ such that $e_i \in y_i$, $i \leqslant m$, only one remains.

By applying the argument above to all events in $\lceil R_1 \rceil$ (and $\lceil R_2 \rceil$), we have that the restriction removes all but one y_i, which is then the maximal configuration in $\lceil R_1 \rceil \mid \lceil R_2 \rceil$. ◀

B.4 Lemmas and Proofs for Sect. 4.3 and Sect. 5

Our goal here is to prove Theorems 36 and 37, and we use the following organization: first (Sect. B.4.1), we show that there is an operational correspondence between R and $\lceil R \rceil$ (Lemma 34), and this requires intermediate lemmas (Lemmas 49, 50, 51) about the encoding of memories and their relation to maximal events. Then (Sect. B.4.2), we use this result to identify isomorphisms of $\mathcal{I}$-structures with label- and order-preserving functions (Lemma 35), and to connect our previous formalism with the encoding of memories (Lemma 39). Finally, Sect. B.4.3 concludes by proving our two main theorems.

B.4.1 Operational Correspondence

▶ **Lemma 49.** *If $R \equiv S$ then $\lceil R \rceil \cong \lceil S \rceil$.*

Proof. Looking back at Definition 13, there are only limited ways for R and S to be structurally equivalent (i.e., in the $\equiv$ relation), and we review them one by one.

In Sum Symmetry, Sum Associativity and α-Conversion, it should be noted that the memory is left untouched, so their encodings are equal.

For Distribution of Memory, we have that $\lceil m \rhd (P \mid Q) \rceil = \lfloor m \rfloor$, and $\lceil (\Upsilon.m \rhd P) \mid (\Upsilon.m \rhd Q) \rceil = \lceil (\Upsilon.m \rhd P) \rceil \mid \lceil (\Upsilon.m \rhd Q) \rceil = \lfloor \Upsilon.m \rfloor \mid \lfloor \Upsilon.m \rfloor = \lfloor m \rfloor \mid \lfloor m \rfloor$. The construction of the isomorphism between $\lfloor m \rfloor$ and $\lfloor m \rfloor \mid \lfloor m \rfloor$ is trivial: let e be an event in $\lfloor m \rfloor$. Note that the only event e' in $\lfloor m \rfloor \mid \lfloor m \rfloor$ such that $\pi_1(e') = e$ is (e, e). Indeed, suppose (e, e'') is

in $\lfloor m \rfloor \mid \lfloor m \rfloor$, for $e'' \neq e$. Then, by Lemma 30, $m(e) \neq m(e'')$, since they are both events in $\lfloor m \rfloor$. But by the definition of parallel composition, (e, e'') should have triggered Err. 1. Similarly, $(e, \star)$ can not be an event in $\lfloor m \rfloor \mid \lfloor m \rfloor$, since it should have triggered Err. 2. Hence, we can map e in $\lfloor m \rfloor$ and (e, e) in $\lfloor m \rfloor \mid \lfloor m \rfloor$ and obtain our isomorphism.

For Scope of Restriction, we have that have $\lceil m \rhd (P \backslash a) \rceil = \lfloor m \rfloor = \lceil (m \rhd P) \backslash a \rceil$. ◀

▶ **Lemma 50.** *The event introduced in the postfixing of a memory event to an identified structure is maximal in the resulting identified structure.*

Proof. The proof is immediate: the event introduced occurs only in the maximal configurations of the identified structure, and hence there cannot be any other event that causes it. ◀

Furthermore, the maximality of an event can be "preserved" by parallel composition:

▶ **Lemma 51.** *For all identified structure $\mathcal{I}_1 = (E_1, C_1, L_1, \ell_1, I_1, m_1)$ with $e_1 \in E_1$ a maximal event in it, and for all identified configuration $\mathcal{I}_2 = (E_2, C_2, L_2, \ell_2, I_2, m_2)$ such that $m_1(e_1) \notin I_2$, $(e_1, \star)$ is maximal in $\mathcal{I}_1 \mid \mathcal{I}_2$.*

Proof. The definition of parallel composition of identified configuration structure (Definition 25) should make it clear that the only event in $\mathcal{I}_1 \mid \mathcal{I}_2$ whose first projection is e_1 is $(e_1, \star)$, since $m_1(e_1) \notin I_2$: all the other pairings of events from E_2 with e_1 resulting in Err. 1. Now, suppose for the sake of contradiction that $(e_1, \star)$ is not maximal in $\mathcal{I}_1 \mid \mathcal{I}_2$. It means that there is a maximal configuration x in $\mathcal{I}_1 \mid \mathcal{I}_2$ and an event $e \in x$ such that $(e_1, \star) <_x e$.

By Definition 5, $\pi_1(x) \in C_1$, and we prove that $\pi_1(x)$ is maximal in C_1 by contradiction. Suppose $\pi_1(x)$ is not maximal in C_1, then there exists $z \in C_1$ such that $\pi_1(x) \subsetneq z$. Assume z is maximal in C_1 (if it is not, then take z' the maximal configuration such that $z \subseteq z'$, which always exists, and note that $e_1 \in z$, since $e_1 \in \pi_1(x)$ and $\pi_1(x) \subset z$. By Stability, since $z \cup \pi_1(x) = z$ is a configuration in C_1, $z \cap \pi_1(x) = z \setminus \pi_1(x)$ also is, and note that for all events e' in $z \setminus \pi_1(x)$, we have that $e_1 \not\leqslant_{z \setminus \pi_1(x)} e'$. Since $e' \in z \setminus \pi_1(x)$, $e' \in z$ and $e_1 \not\leqslant_z e'$, we have that z is a maximal configuration in $\mathcal{I}_1$ where e_1 is not maximal: a contradiction. Hence, we know that $\pi_1(x)$ is maximal in C_1.

Now, we prove that $\pi_1(e) \neq \star$. For the sake of contradiction, suppose that $\pi_1(e) = \star$. Then, observe that the underlying configuration structure $\mathcal{F}(\mathcal{I}_1 \mid \mathcal{I}_2)$ also have this configuration x with $(e_1, \star) <_x e$. Without loss of generality, we can assume that e is a *an immediate cause* of $(e_1, \star)$ [1, Definition 20], that is, that there is no e'' in x such that $(e_1, \star) <_x e'' <_x e$. We can now use [1, Proposition 3] to get that it must be the case that either $\pi_1(e_1, \star) <_{\pi_1(x)} \pi_1(e)$ or $\pi_2(e_1, \star) <_{\pi_2(x)} \pi_2(e)$. But since $\pi_2(e_1, \star) = \star \notin \pi_2(x)$, and since we assumed $\pi_1(e) = \star \notin \pi_1(x)$, both scenario are impossible, so we reached a contradiction, and it must be the case that $\pi_1(e) \neq \star$.

Since $\pi_1(x)$ is maximal in C_1, and since $\pi_1(e), e_1 \in \pi_1(x)$, we reached a contradiction: $\pi_1(e)$ is not caused by e_1, but that cannot be since e_1 is not maximal. Hence, there is no such $e \in x$, and $(e_1, \star)$ is maximal in $\mathcal{I}_1 \mid \mathcal{I}_2$. ◀

▶ **Lemma 34.**

1. *For all transitions $R \xrightarrow{i:\alpha} S$, $\lceil R \rceil \cong \lceil S \rceil \restriction_{\{e\}}$ with e maximal in $\lceil S \rceil$ and $m_S(e) = i$.*
2. *For all transitions $R \overset{i:\alpha}{\rightsquigarrow} S$, $\lceil S \rceil \cong \lceil R \rceil \restriction_{\{e\}}$ with e maximal in $\lceil R \rceil$ and $m_R(e) = i$.*
3. *For any maximal event e in $\lceil R \rceil$, there is a transition $R \overset{m_R(e):\ell_R(e)}{\rightsquigarrow} S$ with $\lceil S \rceil \cong \lceil R \rceil \restriction_{\{e\}}$.*

Proof. We prove the three items separately.

item 1

We proceed by case on α in the transition $R \xrightarrow{i:\alpha} S$:

If $\alpha = a$ For the transition $R \xrightarrow{i:\alpha} S$ to take place, it must be the case that R contains a thread $T = m \rhd a.P + Q$ for some m, P and Q, and that $T \xrightarrow{i:\alpha} \langle i, a, Q\rangle.m \rhd P$. Additionally, the par., res. and $\equiv$ rules of Fig. 4, gives us that it must be the case that

$$R \equiv ((R_{n-1} \cdots ((R_3 \mid ((R_1 \mid T) \mid R_2)\backslash \overrightarrow{b^1}) \mid R_4)\backslash \overrightarrow{b^2} \cdots) \mid R_n)\backslash \overrightarrow{b^m}$$

for some R_i any of which (and its corresponding $\mid$ constructor) could be missing and for some $\overrightarrow{b^j}$, any of which (along with their $\backslash$ constructor) could be missing as well. Hence, the transition can be written as

$$R \xrightarrow{i:a} \underbrace{((R_{n-1} \cdots ((R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash \overrightarrow{b^1}) \mid R_4)\backslash \overrightarrow{b^2} \cdots) \mid R_n)\backslash \overrightarrow{b^m}}_{=S'}$$

with $S' \equiv S$. By Lemma 50, we know that there is a maximal event e_1 in $\lceil \langle i, a, Q\rangle.m \rhd P \rceil$ that has for identifier i. We show that this event can be "traced through" $\lceil S' \rceil$, using four arguments:

- From the par. rule in Fig. 4 we have that $i \notin \mathsf{I}(R_1)$. Using Lemma 51, it follows that there exists a maximal event e_2 in $\lceil R_1 \mid \langle i, a, Q\rangle.m \rhd P \rceil$ such that $\pi_2(e_2) = e_1$, and from Definition 25 that its identifier is i.
- We can use the same reasoning to prove that there is a maximal event e_3 in $\lceil (R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2 \rceil$ such that $\pi_1(e_3) = e_2$ and such that its identifier is i.
- Since $\lceil R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash \overrightarrow{b^1} \rceil = \lceil R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2) \rceil$, it is trivial that e_3 is a maximal event with identifier i in it.
- Using those three arguments repeatedly, and "skipping" them if a parallel composition or a restriction is "missing", we can "trace" the maximal event with identifier i in $((R_{n-1} \cdots ((R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash \overrightarrow{b^1}) \mid R_4)\backslash \overrightarrow{b^2} \cdots) \mid R_n)\backslash \overrightarrow{b^m}$.

But we still need to find a maximal event whose identifier is i *in* $\lceil S \rceil$. Since

$$((R_{n-1} \cdots ((R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash \overrightarrow{b^1}) \mid R_4)\backslash \overrightarrow{b^2} \cdots) \mid R_n)\backslash \overrightarrow{b^m} \equiv S$$

Lemma 49 gives us that $\lceil ((R_{n-1} \cdots ((R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash \overrightarrow{b^1}) \mid R_4)\backslash \overrightarrow{b^2} \cdots) \mid R_n)\backslash \overrightarrow{b^m} \rceil \cong \lceil S \rceil$. Let us write f this isomorphism, we want to prove that $f(e)$ is maximal in $\lceil S \rceil$. To do so, let us use the "if" part of the upcoming Lemma 35 (this part of the lemma is *not* proved using the lemma we are currently proving, but the "and only if" part does, hence the choice of postponing it after this current lemma) and suppose that $f(e)$ is not maximal in $\lceil S \rceil$. Then, there exists a maximal configuration x_m in $\lceil S \rceil$ and an event e' in x_m such that $f(e) <_{x_m} e'$. But since f is an isomorphism, and since f preserves the order in maximal configurations according to the first part of Lemma 35, we have that it must be the case that $e <_{f^{-1}(x_m)} f^{-1}(e')$, which contradicts the maximality of e in $((R_{n-1} \cdots ((R_3 \mid ((R_1 \mid \langle i, a, Q\rangle.m \rhd P) \mid R_2)\backslash \overrightarrow{b^1}) \mid R_4)\backslash \overrightarrow{b^2} \cdots) \mid R_n)\backslash \overrightarrow{b^m}$. Since f is by definition label-preserving, we have that $f(e)$ is a maximal event in $\lceil S \rceil$ such that $m_S(e) = i$. It is obvious that this event is the only one that was added to the encoding of R, so that $\lceil R \rceil = \lceil S \rceil \restriction_{\{e\}}$.

If $\alpha = \tau$ Then it must be the case that R has two threads, $T = m \rhd a.P + Q$ and $T' = m' \rhd \bar{a}.P' + Q'$ for some m, m', a, P, P' and Q, Q', and that

$$R = (\cdots (R_3 \mid (((R_1 \mid T) \mid R_2)\backslash \overrightarrow{b^1} \mid R_4))\backslash \overrightarrow{b^2} \cdots \mid ((R'_1 \mid T') \mid R'_2)\backslash \overrightarrow{c^1} \cdots \mid R_n)\backslash \overrightarrow{b^m}$$

for some R_i, R'_i any of which (and its corresponding $\mid$ constructor) could be missing and for some $\overrightarrow{b^j}, \overrightarrow{c^k}$, any of which (along with their $\backslash$ constructor) could be missing as well. Then the transition becomes

$$R \xrightarrow{i:\tau} (\cdots((R_3 \mid (R_1 \mid (\langle i, a, Q\rangle.m \rhd P) \mid R_2))\backslash\overrightarrow{b^1} \mid R_4)\backslash\overrightarrow{b^2}\cdots \\ (R'_1 \mid ((\langle i, \bar{a}, Q'\rangle.m' \rhd P') \mid R'_2)\backslash\overrightarrow{c^1}\cdots \mid R_n)\backslash\overrightarrow{b^m} = S.$$

We use the same reasoning as in the first case above to deduce that there is a maximal event in $\lceil(\langle i, a, Q\rangle.m \rhd P\rceil$, let us name it e_1, and one in $\lceil\langle i, \bar{a}, Q'\rangle.m' \rhd P'\rceil$, let us name it e'_1, that both have identifier i. Using the same argument as the first case of the proof, we can "trace" in parallel e_1 and e'_1, until the $\mathcal{I}$-structures that hold their "descendants" are put in parallel: at that point, by Definition 25, it should be clear that a single maximal event resulting from their composition will emerge, that it will be labeled τ and have identifier i. Finally, using again the same argument as in the first case, this event resulting from the synchronization of our two maximal events will still be maximal in $\lceil S\rceil$, and hence we can conclude that there exists a maximal event in $\lceil S\rceil$ labeled τ whose identifier is i.

item 2

If $R \overset{i:\alpha}{\rightsquigarrow} S$, then by the Loop Lemma [5, Lemma 6], $S \xrightarrow{i:\alpha} R$. By item 1, we have that $\lceil S\rceil \cong \lceil R\rceil\restriction_{\{e\}}$ with e maximal in $\lceil R\rceil$ and $m_R(e) = i$, which is what we wanted to show.

item 3

For any reachable process R, from Definition 15 and [5, Lemma 10], we have that there exists a *forward-only* trace $O_R \xrightarrow{\star} R$. We consider without loss of generality the following trace: $\emptyset \rhd O_R = R_0 \xrightarrow{1:\alpha_1} R_1 \cdots \xrightarrow{k:\alpha_k} R_k = R$, for some k, $\alpha_1, \ldots, \alpha_k$.

Given two processes S_1, S_2 such that $\lceil S_1\rceil = \lceil S_2\rceil\restriction_{\{e\}}$ for some event e, we write for simplicity that $\lceil S_2\rceil = \lceil S_1\rceil + e$. Using item 1, we have then that $\lceil R_j\rceil = \lceil R_{j-1}\rceil + e_j$, for $j \leqslant k$ and $m(e_j) = j$. Therefore we can construct a bijection h between events e_j in $\lceil R\rceil$ and transitions $t_j : R_{j-1} \xrightarrow{j:\alpha_j} R_j$ with $m(e_j) = j$.

Let e be a maximal event in $\lceil R\rceil$ and let $h(e) = R_{j-1} \xrightarrow{j:\alpha_j} R_j$, where

$$R_0 \xrightarrow{1:\alpha_1} R_1 \cdots R_{j-1} \xrightarrow{j:\alpha_j} R_j \xrightarrow{j+1:\alpha_{j+1}} R_{j+1} \cdots R_{k-1} \xrightarrow{k:\alpha_k} R_k = R$$

As e is maximal, by Definition 33, there exists no event e' in $\lceil R\rceil$ such that $e < e'$. From $\lceil R_{i+1}\rceil = \lceil R_i\rceil + e_{i+1}$, for $i \leqslant k$, we have that $\lceil R_i\rceil \subseteq \lceil R_{i+1}\rceil$ and by induction, $\lceil R_i\rceil \subseteq \lceil R_k\rceil$, for all $i \leqslant k$. Then $\lceil R_{j+i}\rceil \subseteq \lceil R_k\rceil$ and we have that $e_i \not\leqslant e$, for all $j+1 \leqslant i \leqslant k$. It implies that e_i is concurrent with e, for all $j+1 \leqslant i \leqslant k$.

We use here concurrency in a trace and trace equivalence, that have been introduced in the proof of Theorem 22. Consider two consecutive transitions $t_j = R_j \xrightarrow{j+1:\alpha_{j+1}} R_{j+1}$ and $t_{j+1} = R_{j+1} \xrightarrow{j+2:\alpha_{j+2}} R_{j+2}$. Using the bijection h defined above between transitions and events, we have that there exists two event e_j, e_{j+1} in $\lceil R_{j+2}\rceil$ such that $h(e_j) = t_j$ and $h(e_{j+1}) = t_{j+1}$. Suppose that e_j is concurrent to e_{j+1}. By an induction on the structure of memories in $\lceil R_{j+2}\rceil$, we have that $m_{R_j/R_{j+1}} \cap m_{R_{j+1}/R_{j+2}} = \emptyset$ which implies that there exists a transition $t'_j = R_j \xrightarrow{j+2:\alpha_{j+2}} R_{j+2}$ such that t_j is concurrent with t'_j.

Recall that e_i is concurrent with e, for all $j+1 \leqslant i \leqslant k$ in our trace. We can now use the trace equivalence of [5, Definition 9], and use the previous remark repeatedly to re-organize

our original trace as follows:

$$R_0 \xrightarrow{1:\alpha_1} R_1 \cdots R_{j-1} \xrightarrow{j+1:\alpha_{j+1}} R'_{j+1} \cdots R'_{j-1} \xrightarrow{k:\alpha_k} R'_k \xrightarrow{j:\alpha_j} R$$

The core idea being that we can "postpone" the transition that will create the event e in the encoding of R, by flipping it with the other transitions, and have it become the last transition that leads to the same R.

We can use now the Loop Lemma [5, Lemma 6], and have that $R \overset{j:\alpha_j}{\rightsquigarrow} R'_k$. Using item 2 on this transition, we have that $\lceil R'_k \rceil = \lceil R \rceil \restriction_{\{e\}}$. ◀

B.4.2 On Connecting Formalisms

▶ **Lemma 35.** *Letting x_1^m and x_2^m be the unique maximal configuration in $\lceil R_1 \rceil$ and $\lceil R_2 \rceil$, $\lceil R_1 \rceil \cong \lceil R_2 \rceil$ iff there exists a label- and order-preserving bijection between x_1^m and x_2^m.*

Proof. Let $f : \lceil R_1 \rceil \to \lceil R_2 \rceil$ be an isomorphism: using Remark 44, we will take f to be fully determined by its function between events f_E, that we also write f. Then $f : x_1^m \to x_2^m$ is by definition a label-preserving bijection. For two events e, e' in $\lceil R_1 \rceil$, if $e <_{x_1^m} e'$ it follows that for all x_1 in $\lceil R_1 \rceil$ such that $e' \in x_1$ then $e \in x_1$. From f an isomorphism we have that $f^{-1} : \lceil R_2 \rceil \to \lceil R_1 \rceil$ is a well defined morphism. As f^{-1} preserves configurations it follows that for all x_2 in $\lceil R_2 \rceil$, there exists x_1 in $\lceil R_1 \rceil$ such that $f(x_1) = x_2$. Then for all x_2, such that $f(e') \in x_2$ it implies that $e' \in f^{-1}(x_2)$ and $e \in f^{-1}(x_2)$ and finally, $f(e) \in x_2$. Then $f(e) <_{x_2^m} f(e')$, as for all x_2, $x_2 \subseteq x_2^m$. Similarly we proceed for the concurrent events, and obtain that f is an order-preserving bijection on top of being label-preserving.

For the reverse direction, let $f : x_1^m \to x_2^m$ be a label- and order-preserving bijection. All events in $\lceil R_1 \rceil$ are present in x_1^m, and therefore $f : \lceil R_1 \rceil \to \lceil R_2 \rceil$ is a bijection on events. Remains to show that f preserves configurations, that is, for all $x_1 \subseteq x_1^m$, $f(x_1) = \{f(e) \mid e \in x_1\}$ is a configuration in $\lceil R_2 \rceil$. In other words, $f(x_1) = x_2$ must satisfies the properties of Definition 1.

- Finiteness follows from the fact that all configurations in the encoding of an RCCS memory are finite.
- For Coincidence Freeness let us first note that there is a unique and finite maximal configuration x_2^m in which all distinct events are either causal or concurrent. Let us consider $e_2 \neq e'_2$ two events in x_2. Since $e_2 \neq e'_2$ then either (i)$e_2 \leqslant_{x_2^m} e'_2$ (or $e_2 \leqslant_{x_2^m} e'_2$ but this is similar) or (ii)$e_2 \operatorname{co}_{x_2^m} e'_2$. As x_2^m is the unique maximal configuration, the relations above hold for x_2 as well. In the case (i), as $e_2 \neq e'_2$ then $e'_2 \not\leqslant_{x_2} e_2$. We apply the definition of causality (Definition 2) to obtain the configuration $z \subseteq x_2$ where $e_2 \in z$ and $e'_2 \notin z$ as required by Coincidence Freeness. In the case (ii), we use the definition of concurrency, which implies that $e_2 \not\leqslant_{x_2} e'_2$ and $e'_2 \not\leqslant_{x_2} e_2$. Then there exists $z \subseteq x_2$ such that $e_2 \in z$ and $e'_2 \notin z$ and moreover, there exists $z' \subseteq x_2$ such that $e'_2 \in z$ and $e_2 \notin z$.
- To show Finite Completeness, it suffices to note that there exists a unique maximal configuration x_2^m which can be an upper bound for any subset of configurations in $\lceil R_2 \rceil$.
- To show Stability we first note that because there is a single maximal configuration, the condition $\forall x_2, x'_2 \in C_2, x_2 \cup x'_2 \in C_2$ always holds. Then we have to show that for all configurations x_2, x'_2, $x_2 \cap x'_2$ is a configuration.
  We will show that $x_2 \cap x'_2 = f(x_1 \cap x'_1)$ is a configuration in $\lceil R_2 \rceil$ by exploiting the causality and concurrency relations in $x_1 \cap x'_1$. From the second item of Lemma 34 we have that there exists a maximal event e_1^1 in x_1 and from $\lceil S \rceil \restriction_{\{e_1^1\}} \subset \lceil R \rceil$, $x_1 \setminus e_1$ is a configuration in $\lceil R_1 \rceil$. We use the same reasoning for $x_1 \setminus e_1$ in $\lceil S \rceil \restriction_{\{e_1^1\}}$ with e_1^2 maximal

and $\lceil S \rceil\restriction_{\{e_1^1, e_1^2\}} \subset \lceil S \rceil\restriction_{\{e_1^1\}} \subset \lceil R \rceil$. We have then a sequence of events $e_1^1, \cdots e_1^n$ such that $x_1 \setminus e_1, \cdots e_n = x_1 \cap x_1'$.

Let us consider $e_2 \neq e_2'$ two events in x_2 and as f is a bijection, let $e_1 \neq e_1'$ be two events in $f^{-1}(x_2) = x_1$ such that $f^{-1}(e_2) = e_1$ and $f^{-1}(e_2') = e_1'$. If $e_2 \neq e_2'$ then either $e_2 \leqslant_{x_2^m} e_2'$ or $e_2 \operatorname{co}_{x_2^m} e_2'$. It follows that either $e_1 \leqslant_{x_1^m} e_1'$ or $e_1 \operatorname{co}_{x_1^m} e_1'$, respectively, since f^{-1} is label- and order-preserving as well. As x_1^m, x_2^m are the unique maximal configurations the relations above hold for x_2 and x_1 as well.

We therefore have that $f(e_1^1)$ is the maximal event in x_2. Again we use the second item of Lemma 34 and as above we obtain the sequence of events $f(e_1^1), \cdots f(e_1^n)$ such that $f(x_1) \setminus f(e_1), \cdots f(e_n) = f(x_1 \cap x_1')$ is a configuration in $\lceil R_2 \rceil$. ◀

▶ **Lemma 39.** *The underlying configuration structure of $\lceil R \rceil$ is isomorphic to $x_R \downarrow$.*

Proof. Let us abuse the notation by writing $[\![R]\!] = ([\![O_R]\!], x_R)$.

This encoding is based on the following result : as R is reachable there exists a forward-only trace [5, Lemma 10] $\theta : O_R \xrightarrow{i_1:\alpha_1} \cdots \xrightarrow{i_n:\alpha_n} R$, such that for any transition in θ there exists a corresponding event in x [1, Lemma 1].

We reason then by induction on the trace θ. As the base case is similar, we only treat the inductive case.

Suppose that the result holds for R, that is let $h : \mathcal{F}(\lfloor R \rfloor) \to x_R \downarrow$ be an isomorphism. Remember that $\mathcal{F}$ is the identity on events. We use an argument similar to Lemma 35 to show that instead of an isomorphism, we can reason on $h : x^m \to x_R$ as a label- and order-preserving bijection, where x^m is the maximal configuration of $\lfloor R \rfloor$. Indeed, from Definition 3, x_R is the maximal configuration in $x_R \downarrow$ and we can easily derive an identified structure $\mathcal{S}(x_R \downarrow)$ (Lemma 43).

Let $R \xrightarrow{i:\alpha} S$ be an RCCS transition. Then by the operational correspondence between R and $([\![O_R]\!], x)$ [1, Lemma 6] there exists e an event in $[\![O_R]\!]$ such that

$$([\![O_R]\!], x_R) \xrightarrow{e} ([\![O_R]\!], x_R \cup \{e\})$$

with $\ell(e) = \alpha$ and $[\![S]\!] = ([\![O_R]\!], x_R \cup \{e\})$. By the operational correspondence between R and $\lceil R \rceil$ (Lemma 34) there exists a memory event e' such that $\lceil R \rceil = \lceil S \rceil\restriction_{\{e'\}}$ with e' maximal in $\lceil S \rceil$ and $m_S(e') = i$. As there is only one maximal configuration in $\lceil S \rceil$ and as $\lceil R \rceil = \lceil S \rceil\restriction_{\{e'\}}$, it follows that $x^m \cup \{e'\}$ is the maximal configuration in $\lceil S \rceil$.

We have then to show that if $h : x^m \to x_R$ is label- and order-preserving bijection then so is $h' = h \cup \{e \to e'\} : x^m \cup \{e'\} \to x_R \cup e$. We do this by exploiting the correspondence with RCCS and following a reasoning similar to the one in Theorem 37. ◀

B.4.3 Proofs of Theorems 36 and 37

▶ **Theorem 36.** *P_1 and P_2 are* HHPB *(resp.* HPB*) iff $[\![P_1]\!]$ and $[\![P_2]\!]$ are.*

Proof. Let us prove the HHPB case, the other case being similar, and actually simpler.

⇒ Let $\mathcal{R}_{\text{RCCS}}$ be a HHPB between P_1 and P_2 (Definition 32). We show that the following relation

$$\begin{aligned}\mathcal{R} = \{(x_1, x_2, f) \mid\ & x_1 \in [\![P_1]\!], x_2 \in [\![P_2]\!], \exists R_1, R_2 \text{ s.t. } O_{R_1} = P_1, O_{R_2} = P_2, \\ & (R_1, R_2, F) \in \mathcal{R}_{\text{RCCS}} \text{ and } [\![R_1]\!] = ([\![P_1]\!], x_1), [\![R_2]\!] = ([\![P_2]\!], x_2), f = \mathcal{F}(F)\}\end{aligned}$$

is a HHPB between $[\![P_1]\!]$ and $[\![P_2]\!]$.

We first show that for any tuple $(x_1, x_2, f) \in \mathcal{R}$, $f : x_1 \to x_2$ is a label- and order-preserving bijection. For a tuple $(R_1, R_2, F) \in \mathcal{R}_{\text{RCCS}}$ with $F : \lceil R_1 \rceil \to \lceil R_2 \rceil$ an

isomorphism, we have that $F : x_1^m \to x_2^m$, the isomorphism restricted to the maximal configurations x_1^m, x_2^m, with F a label- and order-preserving bijection, by Lemma 35. The functor $\mathcal{F}$ maps isomorphisms on identified structures to isomorphisms on configuration structures, by Lemma 43, and therefore $f = \mathcal{F}(F) : \mathcal{F}(x_1^m) \to \mathcal{F}(x_2^m)$.
Moreover, x_1 and x_2 are the maximal configurations in $x_1\downarrow$ and $x_2\downarrow$, by Definition 3. By Lemma 39, $\mathcal{F}(\lceil R_i \rceil) \cong x_i\downarrow$ which implies that $\mathcal{F}(x_i^m) = x_i$, for $i \in \{1,2\}$. Thus $f = \mathcal{F}(F)$ is well defined and indeed, a label- and order-preserving bijection.
The rest of the proof follows the same structure as in [1, Proposition 6]. Note that $(\emptyset, \emptyset, \emptyset) \in \mathcal{R}$: indeed $(\emptyset \rhd P_1, \emptyset \rhd P_2, \emptyset) \in \mathcal{R}_{\text{RCCS}}$ and $[\![\emptyset \rhd P_i]\!] = ([\![P_i]\!], \emptyset)$, for $i \in \{1,2\}$.
Let us suppose that $(x_1, x_2, f) \in \mathcal{R}$ for $[\![R_i]\!] = ([\![P_i]\!], x_i)$, for $i \in \{1,2\}$ and $f : x_1 \to x_2$ a label- and order-preserving bijection. Note that $\mathcal{F}(\lceil R_i \rceil) \cong x_i\downarrow$, from Lemma 39, and that $\lceil R_i \rceil \cong x_i\downarrow \oplus m$, for some function m, from Definition 24.
To show that $\mathcal{R}$ is a HHPB we have to show that if $x_1 \xrightarrow{e_1} y_1$ (or $x_1 \overset{e_1}{\rightsquigarrow} y_1$) then there exists y_2 such that $x_2 \xrightarrow{e_1} y_2$ (or $x_2 \overset{e_2}{\rightsquigarrow} y_2$ respectively) and such that $(y_1, y_2, f') \in \mathcal{R}$ for some $f' = f \cup \{e_1 \to e_2\}$.
Let $x_1 \xrightarrow{e_1} y_1$, hence by definition, $y_1 = x_1 \cup \{e_1\}$. From the correspondence between RCCS and their encodings (from [1, Lemma 6]), it follows that $R_1 \xrightarrow{i:\alpha} S_1$ such that $[\![S_1]\!] = ([\![P_1]\!], y_1)$.
As $(R_1, R_2, F) \in \mathcal{R}_{\text{RCCS}}$ and as $R_1 \xrightarrow{i:\alpha} S_1$, it follows that there exists a transition $R_2 \xrightarrow{j:\alpha} S_2$ with $F = F'{\restriction}_{\lceil R_1 \rceil}$ and $(S_1, S_2, F') \in \mathcal{R}_{\text{RCCS}}$.
Again from the correspondence between R_2 and $[\![R_2]\!]$ we have that $x_2 \xrightarrow{e_2} y_2$ such that $y_2 = x_2 \cup \{e_2\}$ and $[\![S_2]\!] = ([\![P_2]\!], y_2)$. Then we show that $(y_1, y_2, \mathcal{F}(F')) \in \mathcal{R}$. The only missing argument is that $\mathcal{F}(F') = f \cup \{e_1 \to e_2\}$. Note that, again using Lemma 35, we can consider $F : x_1^m \to x_2^m$ and $F' : y_1^m \to y_2^m$ to be label- and order-preserving bijections on the maximal configurations of $\lceil R_1 \rceil, \lceil R_2 \rceil$ and $\lceil S_1 \rceil, \lceil S_2 \rceil$, respectively. From the definition of postfixing and parallel composition, in Definition 25, we have that $y_1^m = x_1^m \cup \{e_1\}$ and $y_2^m = x_2^m \cup \{e_2\}$. Therefore $F' = F \cup \{e_1 \to e_2\}$ by which we conclude.
We treat similarly the cases where x_2 does a transition, or when the transitions are backwards.

$\Leftarrow$ Let $\mathcal{R}_{\text{CONF}}$ be a HHPB between $[\![P_1]\!]$ and $[\![P_2]\!]$. We show that the following relation

$$\begin{aligned}\mathcal{R} = \{(R_1, R_2, F) \mid\ & O_{R_1} = P_1, O_{R_2} = P_2, [\![R_1]\!] = ([\![P_1]\!], x_1), [\![R_2]\!] = ([\![P_2]\!], x_2),\\ & \text{with } (x_1, x_2, f) \in \mathcal{R}_{\text{CONF}}, F = (f, f_m),\\ & f_m(i) = j, m_1(e) = i, m_2(f(e)) = j, \text{for all } e \in x_1\}\end{aligned}$$

is a HHPB between $\emptyset \rhd P_1$ and $\emptyset \rhd P_2$.
For $(R_1, R_2, F) \in \mathcal{R}$, let us show that F is an isomorphism between $\lceil R_1 \rceil$ and $\lceil R_2 \rceil$. For $f : x_1 \to x_2$ a label- and order-preserving bijection, and for two functions $m_1 : x_1 \to I_1$ (from $\lceil R_1 \rceil$) and $m_2 : x_2 \to I_2$ (from $\lceil R_2 \rceil$) then there exists a unique function $f_m : I_1 \to I_2$ such that $f_m(m_1(e)) = m_2(f_E(e))$, for all $e \in x_1$. This follows from Collision Freeness in the definition of identified structures and from f being a bijection. We write then $F = (f, f_m) : (x_1 \oplus m_1) \to (x_2 \oplus m_1)$. Moreover, as in the first case, we derive that x_1, x_2 are maximal in $\lceil R_1 \rceil, \lceil R_2 \rceil$, respectively. We use Lemma 35 to conclude that $F : \lceil R_1 \rceil \to \lceil R_2 \rceil$ is an isomorphism.
We have that $(\emptyset \rhd P_1, \emptyset \rhd P_2, \emptyset) \in \mathcal{R}$ as $(\emptyset, \emptyset, \emptyset) \in \mathcal{R}_{\text{CONF}}$ and $[\![\emptyset \rhd P_i]\!] = ([\![P_i]\!], \emptyset)$, for $i \in \{1,2\}$.
We suppose now that $(R_1, R_2, F) \in \mathcal{R}$, with $F : \lceil R_1 \rceil \to \lceil R_2 \rceil$. It implies that $(x_1, x_2, f) \in \mathcal{R}_{\text{CONF}}$ for $[\![R_i]\!] = ([\![P_i]\!], x_i)$, $i \in \{1,2\}$ and that $F = (f, f_m)$ with $f : x_1 \to x_2$ label- and

order-preserving.
To show that $\mathcal{R}$ is a HHPB we have to show that if $R_1 \xrightarrow{i:\alpha} S_1$ (or $R_1 \overset{i:\alpha}{\rightsquigarrow} S_1$) then there exists S_2 such that $R_2 \xrightarrow{j:\alpha} S_2$ (or $R_2 \overset{j:\alpha}{\rightsquigarrow} S_2$ respectively) and such that $(S_1, S_2, F') \in \mathcal{R}$ for some F'.
Let $R_1 \xrightarrow{i:\alpha} S_1$. We use again the correspondence between RCCS and their encodings [1, Lemma 6] from which we have that there exists e_1 and $y_1 = x_1 \cup \{e_1\}$ such that $x_1 \xrightarrow{e_1} y_1$ and $[\![S_1]\!] = ([\![P_1]\!], y_1)$. As $(x_1, x_2, f) \in \mathcal{R}_{\text{CONF}}$ it implies that there exists e_2, y_2 and $f' = f \cup \{e_1 \to e_2\}$ such that $x_2 \xrightarrow{e_2} y_2$ and $(y_1, y_2, f') \in \mathcal{R}_{\text{CONF}}$.
Again, from the correspondence between RCCS and configuration structures we have that, from $x_2 \xrightarrow{e_2} y_2$, there exists S_2 such that $R_2 \xrightarrow{j:\alpha} S_2$ with $[\![S_2]\!] = ([\![P_2]\!], y_2)$. From $f' = f \cup \{e_1 \to e_2\}$ it easily follows that $F' = (f', f'_m) = (f \cup \{e_1 \to e_2\}, f_m \cup \{i \to j\})$. We conclude therefore that $(S_1, S_2, F') \in \mathcal{R}$.
Similarly we show the cases where R_1 does a backward transition, or if R_2 does a forward or backward transition. ◀

▶ **Theorem 37.** HHPB *and* B&F bisim *coincide on all CCS processes.*

Proof. Let P_1, P_2 be two processes and let $\mathcal{R}$ be a B&F bisim relation between them as in Definition 20. Then for any $(R_1, R_2, f) \in \mathcal{R}$, R_1, R_2 are two RCCS processes and $f : \mathsf{I}(R_1) \to \mathsf{I}(R_2)$ is a bijection on their identifiers. Letting $\lceil R_i \rceil = (E_i, C_i, L_i, \ell_i, I_i, m_i)$, for $i \in \{1, 2\}$, we show that the relation

$$\mathcal{S} = \big\{(R_1, R_2, F) \mid (R_1, R_2, f) \in \mathcal{R}, F(e_1) = e_2 \iff f(m_1(e_1)) = m_2(e_2)\big\}$$

is a HHPB relation as in Definition 32. We do this by induction on the processes R_1 and R_2, as any process reachable from P_1 has to be in $\mathcal{R}$ (and in $\mathcal{S}$) and vice versa. Remember Remark 44: to define $F : \lceil R_1 \rceil \to \lceil R_2 \rceil = (f_E, f_C, f_m)$, it is enough to define f_E, and we write F for f_E in the following. The base case $(\emptyset \rhd P_1, \emptyset \rhd P_2, \emptyset) \in \mathcal{S}$ is trivial.
Suppose that for $(R_1, R_2, f) \in \mathcal{R}$, we have that $(R_1, R_2, F) \in \mathcal{S}$. Now let $R_1 \xrightarrow{i:\alpha} S_1$ for which we have that $R_2 \xrightarrow{j:\alpha} S_2$ and $g : \mathsf{I}(S_1) \to \mathsf{I}(S_2)$ is a bijection defined as f on $\mathsf{I}(R_1) \subset \mathsf{I}(S_1)$ and $g(i) = j$. Let $e_1 \in \lceil S_1 \rceil$ such that $m_1(e_1) = i$ and $e_2 \in \lceil S_2 \rceil$ such that $m_2(e_2) = j$. Then $G = F \cup \{e_1 \to e_2\}$ is defined by $g(m_1(e_1)) = m_2(e_2)$, and thus $(S_1, S_2, G) \in \mathcal{S}$.
We show now that G is an isomorphism between $\lceil S_1 \rceil$ and $\lceil S_2 \rceil$. G preserves identifiers by definition; it preserves labels as $\ell_1(e_1) = \ell_2(e_2) = \alpha$. Lastly we have to show that it preserves configurations. We can use Lemma 35 to show instead that $G : x_1^m \to x_2^m$ is a label- and order-preserving function, where x_1^m, x_2^m are the maximal configurations in $\lceil S_1 \rceil$ and $\lceil S_2 \rceil$, respectively. Also by Lemma 35 we have that $F : y_1^m \to y_2^m$ is a label- and order-preserving function on y_1^m, y_2^m maximal configurations in $\lceil R_1 \rceil$ and $\lceil R_2 \rceil$, respectively. From Lemma 34, $x_1^m \setminus \{e_1\} \in \lceil R_1 \rceil$ and using Lemma 31 we have that $y_1^m = x_1^m \setminus \{e_1\}$. Therefore we have that $G : x_1^m \to x_2^m$ is a label- and order-preserving function on all events $e \neq e_1$.
Let us suppose, by contradiction, that there exists e such that $e_1 \operatorname{co}_{x_1^m} e$ but that $e_2 \operatorname{co}_{x_2^m} F(e)$ does not hold. Take the maximal event e with such a property. Moreover, let $F(e) = e' <_{x_2^m} e_2$ without loss of generality.
There exists at least one sequence of events $e'_1 \leq \cdots \leqslant e'_n$ such that $e \leqslant_{x_1^m} e'_1$ and such that e'_n is maximal. For simplicity we suppose that there is only one such sequence (the general case uses the same reasoning). From Lemma 34, we have that $S_1 \overset{m(e'_n):\ell(e'_n)}{\rightsquigarrow} S'_1$. We have $F(e) < F(e'_1) < \cdots < F(e'_n)$ and $F(e'_n)$ maximal, since F is label- and order-preserving on all these events. As $(S_1, S_2, g) \in \mathcal{R}$, we have also $(S'_1, S'_2, g') \in \mathcal{R}$, where g'

is defined as g. We apply this reasoning until we reach the process T_1 where e is maximal. Then there exists T_2 such that (T_1, T_2, g'') and g'' is defined as g on the identifiers i, j and $m(e)$.
Then e is maximal however e' is not and we reach a contradiction: from Lemma 34 $S_1 \overset{m(e):\ell(e)}{\rightsquigarrow} S_1'$ but S_2 cannot backtrack on e'. It implies then that $G : x_1^m \to x_2^m$ is a label- and order-preserving function on all events in x_1^m.

- Let P_1, P_2 be two processes and let $\mathcal{R}$ be a HHPB relation between them as defined in Definition 32. Then for any R_1, R_2 two RCCS processes and $f : \lceil R_1 \rceil \to \lceil R_2 \rceil$ an isomorphism such that $(R_1, R_2, f) \in \mathcal{R}$, f is also a bijection on the event identifiers of the memories of R_1, R_2. The relation

$$\mathcal{S} = \big\{(R_1, R_2, F) \mid (R_1, R_2, f) \in \mathcal{R} \text{ and } F(i) = j \iff f(e_1) = e_2, m_1(e_1) = i, m_2(e_2) = j\big\}$$

is a B&F bisim relation as defined in Definition 20. As above, we are using the operational correspondence of Lemma 34. ◀